МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЧИСЛОВІ МЕТОДИ

Комп'ютерний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів,
які навчаються за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології»,
освітньо-професійною програмою «Автоматизація та комп'ютерно-інтегровані технології кібер-енергетичних систем»

Київ
КПІ ім. Ігоря Сікорського
2020

Числові методи: комп’ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології», освітньо-професійної програми «Автоматизація та комп’ютерно-інтегровані технології кібер-енергетичних систем» / КПІ ім. Ігоря Сікорського ; уклад.: О. В., Степанець;. – Електронні текстові дані (1 файл: 3,35 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. – 82 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 18.06.2020 р.) за поданням Вченої ради факультету (протокол № 10 від 25.05.2020 р.)

Електронне мережне навчальне видання

ЧИСЛОВІ МЕТОДИ

Комп’ютерний практикум

Укладач: *Степанець Олександр Васильович, канд.техн.наук, доц.*

Відповідальний редактор: *Волощук Володимир Анатолійович., д-р. техн. наук, доц.*

Рецензент: *Гагарін Олександр Олександрович, канд.техн.наук, доц., КПІ ім. Ігоря Сікорського*

Посібник призначений для студентів, які навчаються за спеціальністю «Автоматизація та комп’ютерно-інтегровані технології», освітньо-професійна програма «Автоматизація та комп’ютерно-інтегровані технології кібер-енергетичних систем», які вивчають курс «Числові методи».

Метою посібника «Числові методи» є набуття практичних навичок застосування числових методів для вирішення типових інженерних задач. Вирішення задач демонструється у сучасному середовищі розрахунків і моделювання Matlab.

ЗМІСТ

Вступ 4
1 Лабораторна робота №1 Робота з векторами та матрицями 5
2 Лабораторна робота №2 Особливості графіки системи Matlab 12
3 Лабораторна робота №3 Пошук коренів алгебраїчних рівнянь 27
4 Лабораторна робота №4 Інтерполяція та апроксимація 42
5 Лабораторна робота №5 Пошук екстремуму функції 53
6 Лабораторна робота №6 Розв'язання звичайних диференційних рівнянь 60
7 Лабораторна робота №7 Основи програмування в Matlab 73
Навчально-методичні матеріали 81
Додаток А. Правила виконання роботи та оформлення протоколів 82

ВСТУП

Сучасний рівень розвитку інформаційних технологій потребує глибоких знань комп'ютерного обладнання та програмного забезпечення. Ефективне використання комп'ютерної техніки дозволяє значно підвищити продуктивність праці, зменшити в декілька разів час на вирішення інженерних задач та підвищити точність розрахунків.

Основний зміст лабораторних робіт присвячений опануванню числових методів вирішення типових інженерних задач, а також отриманню та розвитку навичок роботи з середовищем розрахунків і моделювання Matlab.

У посібнику наведені необхідні теоретичні відомості для вирішення поставлених задач, приклади та прийоми вирішення, висвітлені окремі питання, корисні студентам вказаної спеціальності. Вони можуть бути корисними при виконанні розрахункових робіт із суміжних дисциплін та під час дипломного проектування як додаткова довідкова література.

1 ЛАБОРАТОРНА РОБОТА №1
РОБОТА З ВЕКТОРАМИ ТА МАТРИЦЯМИ

Мета роботи: ознайомитись з правилами і особливостями роботи з векторами та матрицями у середовищі Matlab; отримати навички комп'ютерних розрахунків з використанням матричної алгебри.

1.1 Теоретичні відомості

1.1.1 Способи представлення масивів в MATLAB

У MATLAB всі дані представляються у вигляді масивів. Масиви бувають *одновимірними*, коли використовується один індекс (номер), а можуть бути і багатовимірними (зокрема — двовимірними).

Таким чином, вектор представляється як одновимірний масив, а матриця – як двовимірний. Вектор може представлятися як вектор-стовпець або вектор-рядок, якщо це важливо для обчислень.

Значення елементів вектора-рядка записуються в квадратних дужках через пробіл або кому.

```
>> V1=[1 3 5 7]
```

Значення елементів вектора-стовпця записуються в квадратних дужках через крапку з комою (';').

```
>> V2=[2;1;0;-1]
```

Матриця (двовимірний масив) також записується в квадратних дужках, але як роздільник між рядків може використовуватися ';' або Enter (новий рядок).

```
>> M1=[1 2 4; 4 5 6; 7 8 9]

>> M2 = [5 6 7
2 1 3
7 8 8]
```

1.1.2 Звернення до елементів масиву в середовищі Matlab

Для доступу до окремого елемента *одновимірного* масиву потрібно після його імені вказати в круглих дужках індекс (номер) цього елемента. Наприклад,

перший елемент масиву V1 позначається як V1(1), третій — як V1(1) тощо. Нумерація елементів масиву починається з 1.

Для доступу до окремих елементів двовимірного масиву використовується вираз у круглих дужках, в якому через кому перераховуються його індекси. Першим вказується номер рядка, другим — номер стовпця.

```
>> M1(1,1)
ans=
1
>> M1(2,3)
ans=
6
```

1.1.3 Поелементні операції над векторами

Якщо вектор використовується як аргумент математичних функцій, результатом яких є вектор значень функції, то це означає, що значення функції обчислюються поелементно. В MATLAB для виконання поелементних операцій над векторами призначені операції, наведені в таблиці 3.1. При цьому вектори повинні бути однакової розмірності.

Крапка в MATLAB використовується не тільки для введення десяткових дробів, але і для вказівки того, що операції над масивами однакового розміру повинні виконуватися поелементно.

Таблиця 1.1. Основні поелементні операції над векторами

.+	Поелементне додавання
.-	Поелементне віднімання
.^	Поелементне піднесення до степеня
.*	Поелементне множення
./	Поелементне ділення (першого вектора на другий)
.	Зворотне поелементне ділення (другого вектора на перший)

Нехай маємо вектор

```
>> V3=[2 1 0 -1]
```

При по елементному множенні векторів V1 та V3 матимемо

```
>> V1.*V3
ans=
2  3  0  -7
```

Вектор можна помножити на число. При цьому операція множення виконується поелементно, але крапку перед знаком операції ставити не обов'язково.

```
>>2*V1
ans=
2   6   10   14
```

1.1.4 Операції над матрицями

При використанні матричних операцій слід пам'ятати, що для додавання або віднімання матриці повинні бути одного розміру, а при множенні кількість стовпчиків першої матриці повинно дорівнювати кількості рядків другий матриці. Додавання і віднімання матриць, так само як чисел і векторів, здійснюється за допомогою знаків плюс і мінус.

```
>> M3 = M1+M2
 M3 =
 6 8 11
 6 6 9
14 16 17
```

Для множення матриць призначена зірочка:

```
>> M4=M1*M2
 M4 =
    37 40 45
    72 77 91
 114 122 145
```

Множення матриці на число теж здійснюється за допомогою зірочки, причому множити на число можна як справа, так і зліва:

Транспонування матриці, так само як і вектора, здійснюється за допомогою операції « ' ».

```
>> M5=M1'
M5 =
    1 4 7
    2 5 8
    4 6 9
```

У ряді випадків для роботи необхідно використовувати матриці спеціального вигляду: нульову, одиничну, діагональну тощо. Для цього передбачені окремі функції

Таблиця 1.2. Матриці спеціального вигляду

Функція	Вид матриці	Приклад
zeros	Нульова (розмір задається в параметрах)	Z = zeros(3,4) Z = zeros(6) Z = zeros([3 4])
ones	Одинична прямокутна (одиниці на головній діагоналі)	O = ones(3,4) O = ones(6) O = ones([3 4])
rand	Матриця з випадковими числами, рівномірно розподіленими на інтервалі (0; 1)	Rnd = rand(3,4) Rnd = rand(6) Rnd = rand([3 4])
randn	Матриця з випадковими числами, що розподілені по нормальному закону з нульовим середнім та одиничною дисперсією	Rnd = randn(3,4) Rnd = randn(6) Rnd = randn([3 4])
diag	Діагональна матриця, елементи якої задаються вектором у вхідному параметрі	Створення матриці: D=diag(V1) Виділення головної діагоналі з матриці M1 та формування з неї вектора V5: V5=diag(M1) Виділення k-ї побічної діагоналі з матриці M1 та формування з неї вектора V5: V5=diag(M1,k)

1.1.5 Обернення матриці

Матриця *A* розміру *N*N* називається невиродженою, якщо існує така матриця *B* розміру *N*N*, що

$$A*B=B*A=E$$

де *E* – одинична матриця.

Якщо така матриця *B не* існує, то говорять, що матриця *A* – вироджена.

В Matlab для знаходження оберненої матриці призначена функція *inv* з вхідним аргументом – матрицею.

Наприклад, знайдемо матрицю M6, що обернена до матриці M1:

```
>> M6=inv(M1)
M6=
1.0  -4.667  2.666
-2.0  6.333  -3.333
1.0  -1.999 1.0
```

Правильність отриманого результату можна перевірити, перемноживши матриці M1 та M6:

```
 >> M1*M6
ans =
1.0000   0        0
0.0000   1.0000   0.0000
0.0000   0.0000   1.0000
```

1.1.6 Знаходження визначника матриці

Визначник квадратної матриці - це скалярна величина (дійсне число). За значенням визначника можна судити про те, чи є матриця виродженою чи ні. Так, якщо визначник рівний 0, то матриця вироджена.

Для знаходження визначника в Matlab використовується функція *det.*

Знайдемо визначник матриці M1 і відразу запишемо його значення в змінну detM1:

```
>> detM1=det(M1)
     ans=
     -3
```

1.2 Порядок виконання роботи

Згідно з номером варіанту виконати наступні завдання.

1.1. Сформувати матриці A , B , C згідно наведених шаблонів, де N – номер варіанту.

$$A=\begin{bmatrix} 1 & 0.75 & 1.32 \\ 2 & 3 & 5 \\ 1.23 & 0.2N & 2.1N \end{bmatrix}, B=\begin{bmatrix} 1.2N & 0.75 & 0.87 \\ 1 & 1.27 & 0.53N \\ 3 & 6.1 & 4 \end{bmatrix}, C=\begin{bmatrix} 1.2N & -2.57 & 0.7 \\ 2.45 & 1.27N & 0.32N \\ 1.11N & 6.1 & 3.47 \end{bmatrix}$$

.

1.2 Знайти матричні вирази відповідно до таблиці

№	Матричний вираз	№	Матричний вираз
1	$2AA^T-(CB)^2+2B$	13	$2AA^T-(CB)^2+2A$
2	A^3-A^2+ABC	14	A^3-A^2-ACB
3	$AC-3C^TB^T-2B$	15	$AC-3C^TB^T-1.6A$

№	Матричний вираз	№	Матричний вираз
4	$(AB^T - C)(A + BB^T)$	16	$(AB^T - C)(C + AB^T)$
5	$CBA^2 - 3BC + 2CB$	17	$CBA^2 - ABC + 2CB$
6	$(A^3 - BAC) * A$	18	$(A^3 - BAC) / A$
7	$(A^3 + CB)^T (A - B^2C)$	19	$(A^3 + CB)^T (A^2 - BC)$
8	$(BB^T + C^TC)C^T - 3A$	20	$C(BB^T + C^TC)C^T - 3A$
9	$-3C^TCAB + AB^TA$	21	$-3C^TCAB + BB^TA$
10	$(BCB - ACA)^T / A$	22	$(BCB - ACA)^T A$
11	$(A^2 - ACB) / A^2$	23	$(A^2 - ACB) / A^4$
12	$2A^3 + A^2 + ABC$	24	$A^4 + A^3 + A^2 + ABC$

1.3. Обчислити значення функції f(x) для усіх елементів матриці A та сформувати результат у вигляді нової матриці того ж виміру, що і матриця A (див. таблицю).

№	Функція	Матриця A
1	$f(x) = x^{0.5} + \sin(x) + 0.1x$	$\begin{bmatrix} 0.1 & 2 & 5 \\ 1.5 & 3.75 & 6.4 \end{bmatrix}$
2	$f(x) = \sqrt{x} + \cos(x) + 0.2x$	$\begin{bmatrix} 0.5 & 1.98 & 4.23 \\ 2.5 & 3 & 5.41 \end{bmatrix}$
3	$f(x) = \sin(x) + \cos(2x) + 0.3x$	$\begin{bmatrix} 0.88 & 3.8 & 5.1 \\ 2.5 & 4.2 & 6.05 \end{bmatrix}$
4	$f(x) = \cos(x) + \sin(2x) + 0.4x$	$\begin{bmatrix} 0.1 & 2 & 5 \\ 1.5 & 3.75 & 6.4 \end{bmatrix}$
5	$f(x) = \sin(x) + 0.5\cos(2x) + 0.5x$	$\begin{bmatrix} 0.5 & 1.98 & 4.23 \\ 2.5 & 3 & 5.41 \end{bmatrix}$
6	$f(x) = \cos(x) + 0.5\sin(2x) + 0.6x$	$\begin{bmatrix} 0.88 & 3.8 & 5.1 \\ 2.5 & 4.2 & 6.05 \end{bmatrix}$
7	$f(x) = e^{-0.3x}\sin(x) + 0.7x$	$\begin{bmatrix} 0.1 & 2 & 5 \\ 1.5 & 3.75 & 6.4 \end{bmatrix}$
8	$f(x) = e^{-0.3x}\cos(x) + 0.8x$	$\begin{bmatrix} 0.5 & 1.98 & 4.23 \\ 2.5 & 3 & 5.41 \end{bmatrix}$
9	$f(x) = x^2\cos(2x) + 0.9x$	$\begin{bmatrix} 0.88 & 3.8 & 5.1 \\ 2.5 & 4.2 & 6.05 \end{bmatrix}$

№	Функція	Матриця А
10	$f(x)=\sqrt{x}+\sin(3x)+x$	$\begin{bmatrix}0.1 & 2 & 5\\ 1.5 & 3.75 & 6.4\end{bmatrix}$
11	$f(x)=\sqrt{x}+\cos(1.8x)+1.1x$	$\begin{bmatrix}0.5 & 1.98 & 4.23\\ 2.5 & 3 & 5.41\end{bmatrix}$
12	$f(x)=x\cdot e^{-x}\sin(0.8x)+1.3x$	$\begin{bmatrix}0.88 & 3.8 & 5.1\\ 2.5 & 4.2 & 6.05\end{bmatrix}$
13	$f(x)=e^{-\sin(x)}+1.4x$	$\begin{bmatrix}0.1 & 2 & 5\\ 1.5 & 3.75 & 6.4\end{bmatrix}$
14	$f(x)=e^{-\cos(x)}+1.5x$	$\begin{bmatrix}0.5 & 1.98 & 4.23\\ 2.5 & 3 & 5.41\end{bmatrix}$
15	$f(x)=\ln(x)+1.1\sin(x)+1.6x$	$\begin{bmatrix}0.88 & 3.8 & 5.1\\ 2.5 & 4.2 & 6.05\end{bmatrix}$
16	$f(x)=\ln(x)+0.9\cos(x)+1.7x$	$\begin{bmatrix}0.1 & 2 & 5\\ 1.5 & 3.75 & 6.4\end{bmatrix}$
17	$f(x)=\ln(x)\sin(2x)+1.8x$	$\begin{bmatrix}0.5 & 1.98 & 4.23\\ 2.5 & 3 & 5.41\end{bmatrix}$
18	$f(x)=\ln(0.86x)\cos(0.75x)+1.9x$	$\begin{bmatrix}0.88 & 3.8 & 5.1\\ 2.5 & 4.2 & 6.05\end{bmatrix}$
19	$f(x)=1/x+\sin(0.56x)+2x$	$\begin{bmatrix}0.1 & 2 & 5\\ 1.5 & 3.75 & 6.4\end{bmatrix}$
20	$f(x)=1.23/x+\cos(1.44x)+2.1x$	$\begin{bmatrix}0.5 & 1.98 & 4.23\\ 2.5 & 3 & 5.41\end{bmatrix}$

1.3 Контрольні запитання

1. Як сформувати у робочому просторі вектор-стовпець і вектор-рядок?
2. Як виконувати прості векторні операції у системі MATLAB? Які особливості зустрічаються при використанні операторів для виконання елементарних операцій над векторами?
3. За допомогою якого оператора проводяться операції індексації елементів матриці?
4. Як провести індексацію елементів матриці за допомогою вектора?
5. Назвіть способи формування матриць в робочому середовищі MATLAB.
6. Які команди у системі MATLAB існують для створення стандартних (одинична, діагональна і т.д.) матриць?

2 ЛАБОРАТОРНА РОБОТА №2
ОСОБЛИВОСТІ ГРАФІКИ СИСТЕМИ MATLAB

Мета роботи: ознайомитись зі способами графічного представлення результатів розрахунків; отримати навички використання графічних елементів Matlab, оформлення результатів.

2.1 Теоретичні відомості

Система MATLAB володіє широким набором засобів візуалізації для побудови графіків функцій однієї і двох змінних і відображення різних типів даних. Всі графіки виводяться в графічні вікна. Вид графіків визначається аргументами графічних команд і може бути змінений за допомогою інструментів графічного вікна.

2.1.1 Побудова двовимірної графіки

Matlab має ряд інструментів для візуалізації графіків функції однієї змінної. Використовуючи різні функції побудови графіків залежно від обставин, можна досягти якнайкращого з позиції сприйняття результату.

Загальний порядок побудови графіків наступний:

1. підготовка даних для графіків (діапазон незалежної змінної з проміжними точками, значення залежної змінної у цих точках);
2. вибір виду графіка, що відповідає поставленій задачі, та власне побудова;
3. оформлення графіка для подальшого використання (назва графіка, підписи осей, легенда тощо).

Розглянемо побудову графіка на прикладі синусоїди у діапазоні [0;10].

Достатньо універсальна графічна функція реалізується командою *plot*. У найпростішому випадку вона викликається з двома вхідними аргументами – парою *x* та *y* (тобто *plot* виводить залежність компонент одного вектора від компонентів іншого).

```
>> x=0:0.1:10; %вектор значень аргументу
>> y=sin(x); %вектор значень шуканої функції у точках, визначених вектором аргументів x
>> plot(x,y) % функція побудови графіка функції по отриманим векторам
```

У результаті з’явиться графічне вікно «Figure 1» з отриманим графіком

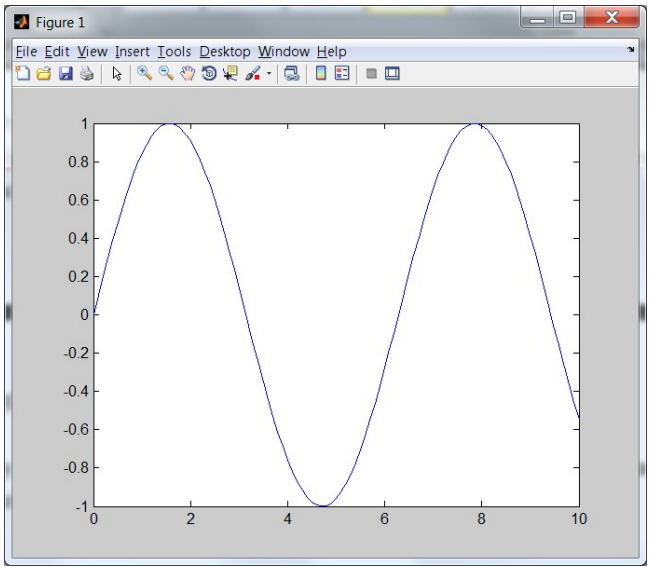

Рис. 2.1. Вікно з графіком функції

Для того, щоб покращити читання графіку, доречно нанести на нього координатну сітку. Наявність сітки регулюється командою «grid»:

`grid on` — побудова сітки;

`grid off` — вимкнення сітки.

Крок сітки у системі MATLAB відповідає цілим одиницям виміру, щоб полегшити сприймання графіка.

Ввівши в командну строку команду

```
>> grid on
```

отримаємо наступний результат:

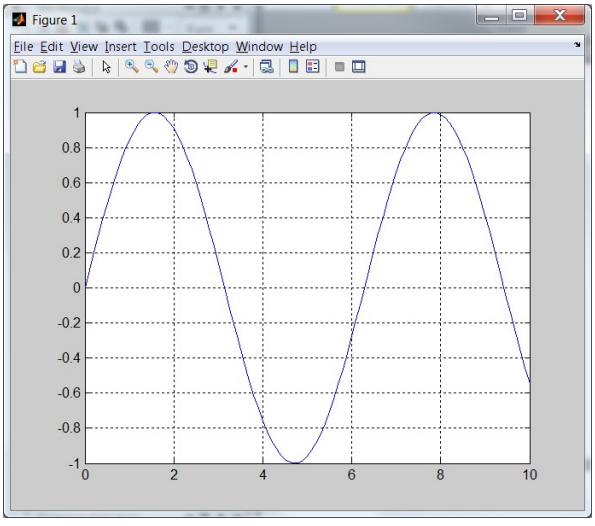

Рис. 2.2. Вікно з графіком функції та координатною сіткою

Для наочнішого зображення графіка функції, особливо у випадку побудови кількох графіків в одному вікні, використовуються допоміжні

параметри, що визначають тип лінії, колір і маркери. Ці параметри визначаються значенням третього (після *x*, *y*) додаткового параметру функції *plot*. Параметр записується в апострофах, наприклад, виклик

```
>> plot(x, y,'go--')
```

приводить до побудови графіка зеленою штриховою лінією, яка розмічена круглими маркерами.

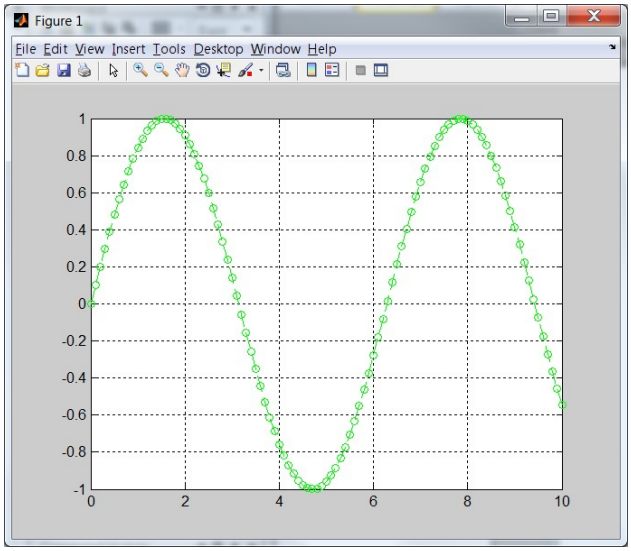

Рис. 2.3. Графік функції з оформленням лінії

Всього в додатковому аргументі може бути заповнено три позиції, що визначають колір, тип маркерів і стиль лінії. Позначення для них наведені в таблиці. Послідовність запису параметрів, може бути довільною, допустимо вказувати не всі параметри.

Таблиця 2.1. Оформлення графіків

Колір лінії		Маркер лінії		Тип лінії	
Позначення	Опис	Позначення	Опис	Позначення	Опис
R	Червоний	.	Точка	-	Суцільна
Y	Жовтий	o	Коло	--	Пунктирна
G	Зелений	x	Хрест	-.	Штрих-пунктир
C	Блакитний	*	Зірочка		
B	Синій	+	Плюс	;	Подвійний пунктир
M	Фіолетовий	s	Квадрат		
K	Чорний	d	Ромб		
W	Білий	V	Трикутник		
		A	Трикутник		
		<	Трикутник		
		>	Трикутник		
		P	П’ятикутник		
		H	Шестикутник		

Команда *plot* дозволяє відображати графіки декількох функцій в одних координатних осях. Спочатку необхідно обчислити значення аргументів та відповідних їх функцій, а потім викликати команду *plot*, після якої в дужках вказуються через кому пари «аргумент-функція», і, при бажанні, стиль кожної з ліній графіків функцій.

Допускається побудова довільного числа графіків функцій, стилі всіх ліній можуть бути різними. Крім того, області побудови кожної з функцій не обов'язково повинні співпадати, але тоді слід використовувати різні вектори для визначення значень аргументів і обчислювати значення функцій від відповідних векторів.

```
>> x1=0:0.1:10; %вектор аргументів 1-го графіка
>> x2=2:0.1:12; %вектор аргументів 2-го графіка
>> y1=sin(x1); %вектор функції 1-го графіка
>> y2=cos(x2); %вектор функції 2-го графіка
>>  plot(x1,y1,'rx-.',x2,y2,'bH--')  % команда побудови графіків
>> grid on % ввімкнення координатної сітки
```

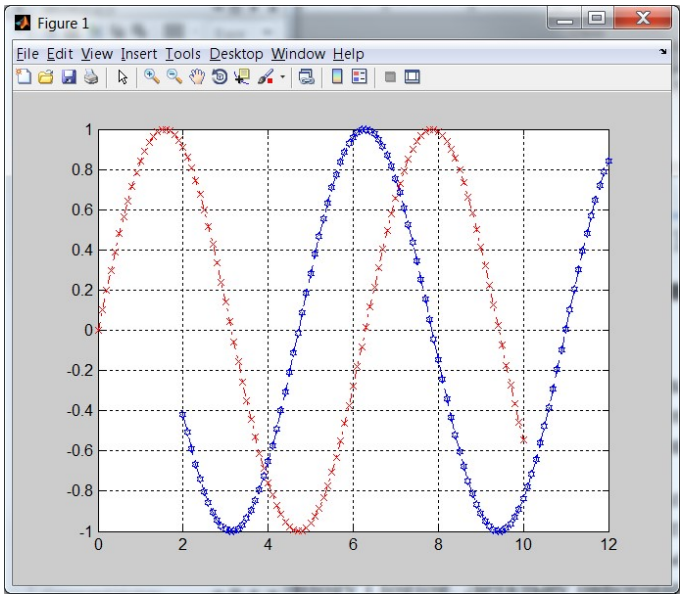

Рис. 2.4. Два графіки в одній площині побудови

Для функції з невизначеним наперед набором значень аргументу можна використати команду *fplot* – аналог команди *plot*, але з автоматичним підбором кроку при побудові графіка. Першим вхідним параметром команди *fplot* є ім'я файл-функції (виділяється апострофами), а другим – вектор, елементами якого є границі відрізка побудови:

```
fplot('ім' я файл-функції' [a,b]).

>>fplot('sin(x)',[0,10])
```

Ця функція також дозволяє побудувати графіків таких функцій, як *sin(x)/x*, для яких мають місце усунені невизначеності. В таких випадках використовується графічна команда у вигляді:

fplot('f(x)', [xmin xmax], n),

де *f(x)*–ім'я функції, [*xmin xmax*] – інтервал побудови графіку, *n* –кількість частин, на які розбивається інтервал (за замовчуванням *n*=25).

Ця команда дозволяє будувати функцію, задану в символьному вигляді, в інтервалі зміни аргументу *x* від значення *xmin* до значення *xmax* без фіксованого кроку зміни *x*. Незважаючи на те, що в процесі обчислень виводиться попередження про помилку (ділення на 0), але графік будується правильно, при *x=0 sinx/x=l*.

У системі MATLAB є функція *comet*, яка дозволяє простежити за рухом точки вздовж траєкторії параметрично заданої лінії. Виклик команди

comet(x,y) (для трьохвимірного графіка – *comet3(x,y,t)*)

приводить до появи графічного вікна, на вісях якого зображується переміщення точки у вигляді руху комети з хвостом. Управління швидкістю руху здійснюється зміною кроку при визначенні вектора значень параметру.

Для відображення графіків в логарифмічному і частково логарифмічному масштабах призначені спеціальні команди:

loglog – використовується логарифмічний масштаб по обох вісях;

semilogx – використовується логарифмічний масштаб тільки по вісі абсцис;

semilogy – використовується логарифмічний масштаб тільки по вісі ординат.

Вхідні параметри графічної команди задаються так само, як і при використовуванні команди *plot*.

Якщо дві функції мають непорівнянні значення та їх відображення в одному вікні неможливе, то для порівняння цих функцій треба застосовувати команду *plotyy*. Команда *plotyy* викликається від двох пар вхідних параметрів (векторів) і приводить до появи двох ліній графіків, кожному з яких відповідають власні осі координат.

Побудувати графік у полярних координатах можна за допомогою функції polar. Наприклад, якщо потрібно побудувати графік функції $r = sin(3\varphi)$ в полярних координатах, потрібно виконати наступні дії:

```
>> phi = 0 : 0.01 : 2*pi; %вектор аргумента в заданих межах
>> r = sin( 3* phi); %вектор значення функції
```

```
>> polar(phi , r) %побудова графіка
```

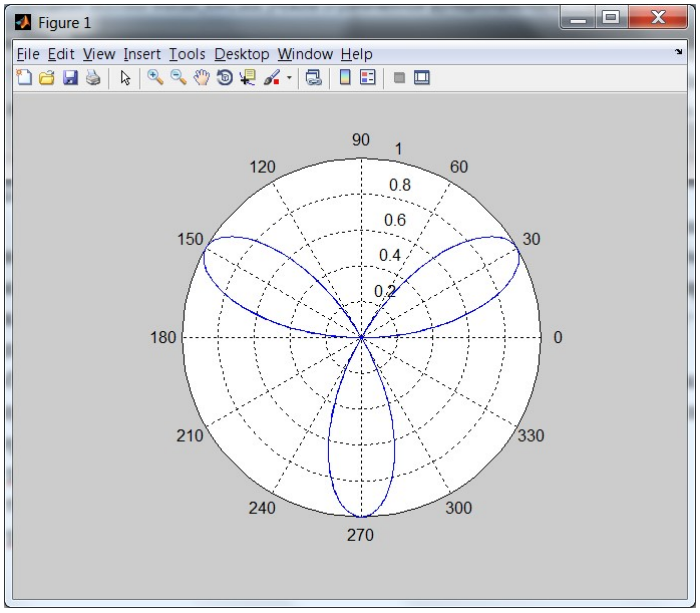

Рис. 2.5. Графік функції в полярній системі координат

2.1.2 Оформлення графіків

Для оформлення графіків (формування підписів тощо) в MATLAB існують спеціальні команди і функції.

Заголовок розміщується в графічному вікні за допомогою команди *title*, вхідним параметром якої є рядок, записаний в апострофах.

За наявності декількох графіків вимагається розташувати легенду, яка формується командою *legend*. Написи легенди, укладені в апострофи, вказуються у вхідних параметрах команди *legend*, їх число повинне співпадати з числом ліній графіків. Крім того, останній додатковий вхідний параметр визначає положення легенди на рисунку (приймає значення від 0 до 4).

Команди *xlabel* і *ylabel* призначені для формування підписів до осей, їх вхідні параметри також записуються в апострофах.

```
>> x1=0:0.1:10;
>> x2=2:0.1:12;
>> y1=sin(x1);
>> y2=cos(x2);
>> plot(x1,y1,'rx-.',x2,y2,'bH--')
>> grid on
>> title('Графіки функцій')
>> xlabel('x')
>> ylabel('y')
```

```
>> legend('sin(x)','cos(x)',0)
```

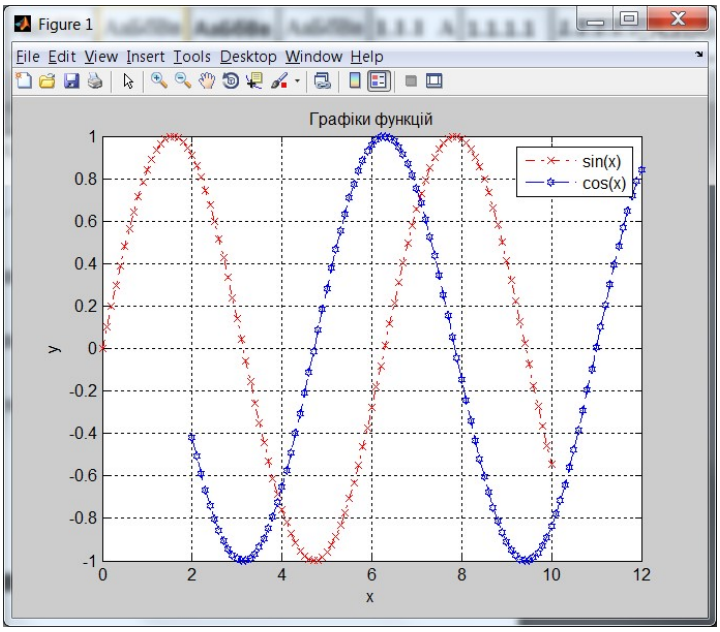

Рис. 2.6. Оформлені графіки функцій з необхідними написами

Користувач може самостійно задавати режим масштабування. Для цього треба вибирати наступні команди:

axis ([xmin xmax ymin ymax]) – здійснює явне завдання діапазону змінних по вісям координат;

axis square – здійснює завдання однакових діапазонів змінних по обох вісях координат;

axis equal – здійснює завдання однакового масштабу по обох вісях координат;

axis auto – здійснює повернення до масштабу, встановленому за замовчуванням;

zoom on – здійснює включення режиму інтерактивної зміни масштабу для поточного графіка, при якому масштаб може змінюватись за допомогою миші;

zoom off – здійснює виключення режиму інтерактивної зміни масштабу.

2.1.3 Виведення кількох графіків

MATLAB показує графічні об’єкти в спеціальних графічних вікнах, що мають в заголовку слово Figure (зображення, зовнішній вигляд, фігура).

Якщо ми, не прибираючи з екрану дисплея перше графічне вікно, вводимо і виконуємо ще один набір команд, то отримуємо новий графік функції в тому ж самому графічному вікні (при цьому старі вісі координат і графік в ньому пропадають – також цього можна досягти командою *clf*, а командою *cla*

видаляють лише графік з приведенням осей координат до їх стандартних діапазонів від 0 до 1):

Якщо потрібно інший графік провести «поверх першого графіка», то перед виконанням другої графічної команди *plot*, потрібно виконати команду:

```
>> hold on
```

яка призначена для утримання поточного графічного вікна. В результаті буде отримано наступне зображення:

Того ж самого можна добитися, вимагаючи від функції *plot* побудувати відразу декілька графіків в рамках одних і тих же осей координат, як показано раніше.

До недоліків вказаних способів побудови декількох графіків в межах одних і тих же осей координат відноситься використання одного і того ж діапазону зміни координат, що при непорівнянних значеннях двох функцій приведе до поганого зображення графіка однієї з них.

Якщо все ж потрібно одночасно візуалізувати декілька графіків так, щоб вони не заважали один одному, то це можна зробити двома способами. По-перше, можна побудувати їх в різних графічних вікнах. Наприклад, побудувавши графіки функцій *sin* і *cos* в межах одного графічного вікна, обчислюємо значення для функції *exp*:

```
>> y3=exp(x1);
```

Після цього виконуємо команди:

```
>> figure;
>> plot(x1,y3);
```

які побудують графік функції *exp* в новому графічному вікні, оскільки команда figure створює нове (додаткове) графічне вікно, і всі подальші за нею команди побудови графіків виводять їх в нове вікно.

Другим рішенням даної задачі показу відразу декількох графіків без конфлікту діапазонів осей координат є використання функції *subplot*. Ця функція дозволяє розбити область виведення графічної інформації на декілька частин, в кожну з яких можна вивести графіки різних функцій.

Наприклад, для раніше виконаних обчислень з функціями *sin, cos* і *exp*, будуємо графіки перших двох функцій в одній області, а графік третьої функції – в другій області одного й того ж графічного вікна:

```
>> subplot(1,2,1)
>> plot(x1,y1,'rx-.',x2,y2,'bH--')
>> subplot(1,2,2)
```

```
>> plot(x1,y3)
>> grid on
```

внаслідок чого отримуємо графічне вікно наступного вигляду:

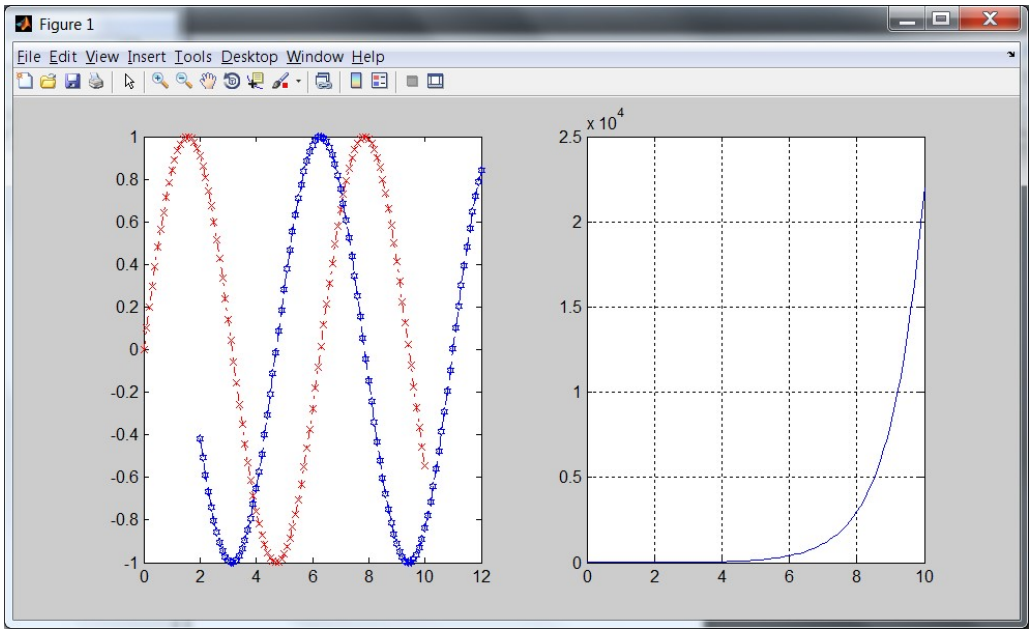

Рис. 2.7. Графіки в різних площинах в одному графічному вікні

Функція *subplot* приймає три числові аргументи, перший з яких дорівнює числу рядів графічної області, друге число дорівнює числу колонок цієї області, а третє число – номеру частини (номер відлічується як в матриці).

Описані варіанти дійсні й для тривимірних графіків.

2.1.4 Тривимірна графіка

Графік функції двох змінних в MATLAB – це поверхня, розташована над областями визначення функції. Тому для побудови такого графіка потрібне використання тривимірного зображення. Для зображення одновимірної кривої в тривимірному просторі використовується команда

plot3(X,Y, Z),

де *X, Y* і *Z* – матриці із значеннями функції (точками *z*) в наборах (*x,y*).

У системі MATLAB є спеціальна функція для здобуття двовимірних масивів *X* і *Y* по одновимірних масивах *x, y*.

Нехай потрібно відобразити поверхню, що описується рівнянням

$z = e^{-x^2-y^2}$, де $x \in [-3;3]$, $y \in [-2;2]$.

Для здобуття матриць *X* і *Y*, що містять значення точок в цій прямокутній сітці, використовується функція:

[X,Y]=meshgrid(x,y);

Обчислимо тепер на отриманій прямокутній сітці значення функції *z*:

Z = exp(- X.^2 -Y.^2)

Тепер застосуємо функцію *plot3*, яка була описана вище, і отримаємо наступний графік:

```
>> x=-3:0.1:3;
>> y=-2:0.1:2;
>> [X,Y]=meshgrid(x,y);
>> Z = exp( - X.^2 -Y.^2 )
>> plot3(X,Y,Z)
>> grid on
```

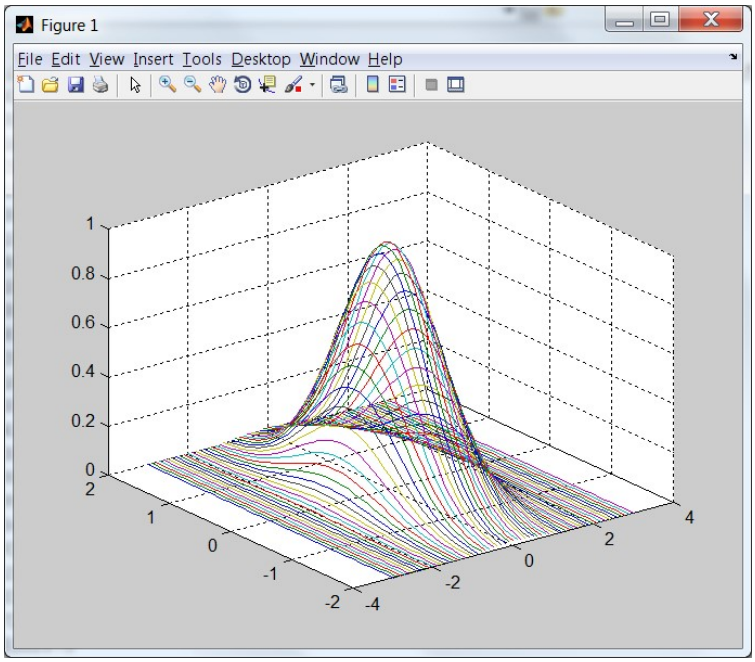

Рис. 2.8. Тривимірний графік, побудований функцією *plot3*

Щоб побудувати тривимірні лінії, задані параметрично, застосовується інша форма виклику функції *plot3*:

plot3(x, y, z)

тут *x, y* і *z* – одновимірні масиви координат точок, які треба послідовно з'єднати відрізками прямих.

Наприклад,

```
>> t=0:.1:50;
>> x=0.5*t. *cos(t);
>> y=0.5*t.*sin(t);
>> z=0.25*t;
>> plot3(x,y,z);
```

```
>> grid on
```

Призведе до утворення наступного графіка:

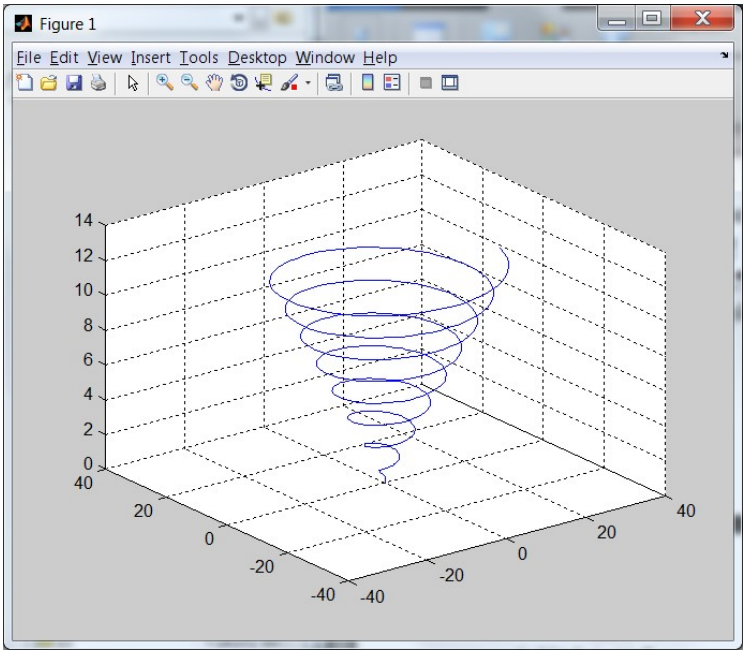

Рис. 2.9. Тривимірний графік функції, заданої параметрично

Слід зазначити, що функції по обробці графіків, допустимі в двовимірному випадку, працюють і для тривимірних зображень. Окрім цієї простої функції побудови графіків в MATLAB є набір інструментів, що дозволяє зробити об'єкти, що відображуються, наочнішими. Це функції *mesh*, *surf* і *surfl*.

Функція *mesh* сполучає обчислені сусідні точки поверхні графіка відрізками прямих і показує в графічному вікні системи MATLAB плоску проекцію такого об'ємного «каркасно-ребристого» тіла.

Mesh сполучає сусідні обчислені точки відрізками, причому невидимі лінії при відображенні ховаються. Якщо ж такі лінії для відображення необхідні, потрібно скористатися командою:

```
>> hidden off
```

Для раніше розглянутого прикладу зображення $z = e^{-x^2-y^2}$ використання функції *mesh* дасть наступний результат:

```
>> mesh(X,Y,Z)
```

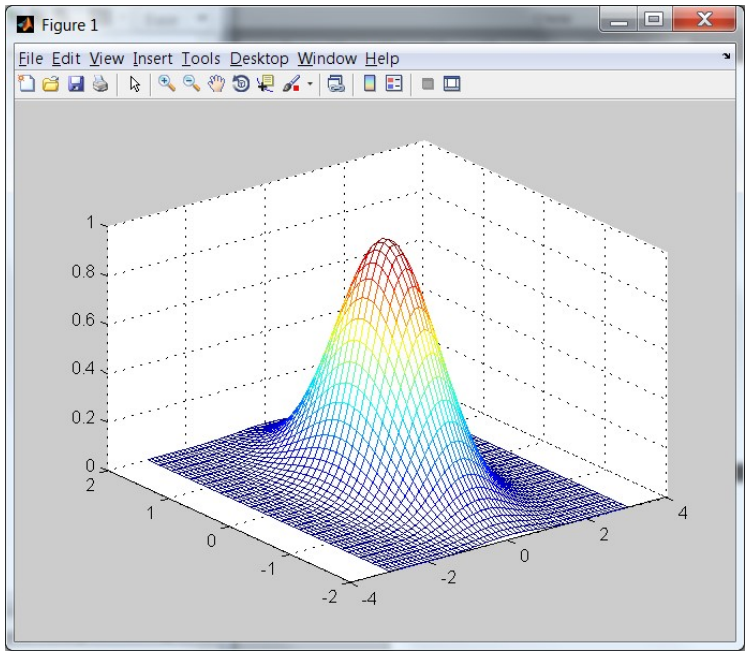

Рис. 2.10. Тривимірний графік, побудований функцією mesh

Існують й інші функції, що візуалізують функції двох змінних.

- *surf* – команда виконує заповнення кольором комірок каркасної поверхні;
- *meshc, surfc* – команди формують кольорові поверхні разом з лініями рівня на площині x0y;
- *contour* – команда формує площинний графік з лініями рівня;
- *contourf* – команда формує залитий кольором плоский графік з лініями рівня;
- *contour3* – команда відображає поверхню, яка сформована з ліній рівня;
- *surfl* – команда формує освітлену поверхню.

Всі перераховані команди допускають таку ж форму завдання вхідних параметрів, що і команда *mesh*.

Команда *shading interp* прибирає лінії сітки з поверхні, побудованої сімейством функцій surf, заповнюючи її кольором.

Команда *colorbar* приводить до відображення в графічному вікні стовпчика, що показує співвідношення між кольором і значенням функції *z(x, y)*. Кольори палітри графіка можна змінювати за допомогою команди *colormap*. Наприклад, *colormap(gray)* відображає графік у відтінках сірого кольору. Назви кольорів, які задають вид палітри, та операції зміни кольорів, їх відтінків, наведені нижче:

- *bone* – схожа на палітру gray, але з легким відтінком синього кольору;
- *colorcube* – задає тон кольору, який змінюється від темного до яскравого;
- *cool* – задає відтінки голубого і пурпурного кольорів;
- *copper* – задає відтінки мідного кольору;

- *hot* – задає плавну зміну тону кольору;
- *hsv* – задає плавну зміну тонів барв веселки;
- *jet* – задає плавну зміну тонів у послідовності: синій–блакитний–зелений–жовтий–червоний;
- *spring* – задає відтінки пурпурового і жовтого;
- *summer* – задає відтінки зеленого і жовтого;
- *winter* – задає відтінки синього і зеленого.

2.2 Порядок виконання роботи

Завдання виконуються згідно отриманого варіанту.

2.1 Побудувати графіки функцій однієї змінної на вказаних інтервалах. Вивести графіки різними способами:

- в окремі графічні вікна;
- в одне вікно на одні вісі;
- в одне вікно на окремі вісі.

Дати заголовки, розмістити підписи до осей, легенду, рекомендується використовувати різні кольори, стилі ліній і типи маркерів, нанести сітку.

№	Функція	Інтервал побудови
1	$f(x) = x^{0.5} + \sin(x) + 0.1x$	$x \in [0{,}1; 10{,}1]$
2	$f(x) = \sqrt{x} + \cos(x) + 0.2x$	$x \in [0{,}5; 8]$
3	$f(x) = \sin(x) + \cos(2x) + 0.3x$	$x \in [-1{,}5; 9{,}1]$
4	$f(x) = \cos(x) + \sin(2x) + 0.4x$	$x \in [-2{,}1; 7{,}5]$
5	$f(x) = \sin(x) + 0.5\cos(2x) + 0.5x$	$x \in [0{,}3; 8{,}7]$
6	$f(x) = \cos(x) + 0.5\sin(2x) + 0.6x$	$x \in [0{,}1; 10{,}1]$
7	$f(x) = e^{-0.3x}\sin(x) + 0.7x$	$x \in [0{,}5; 8]$
8	$f(x) = e^{-0.3x}\cos(x) + 0.8x$	$x \in [-1{,}5; 9{,}1]$
9	$f(x) = x^2\cos(2x) + 0.9x$	$x \in [-2{,}1; 7{,}5]$
10	$f(x) = \sqrt{x} + \sin(3x) + x$	$x \in [0{,}3; 8{,}7]$
11	$f(x) = \sqrt{x} + \cos(1.8x) + 1.1x$	$x \in [0{,}1; 10{,}1]$
12	$f(x) = x \cdot e^{-x}\sin(0.8x) + 1.3x$	$x \in [0{,}5; 8]$
13	$f(x) = e^{-\sin(x)} + 1.4x$	$x \in [-1{,}5; 9{,}1]$
14	$f(x) = e^{-\cos(x)} + 1.5x$	$x \in [-2{,}1; 7{,}5]$
15	$f(x) = \ln(x) + 1.1\sin(x) + 1.6x$	$x \in [0{,}3; 8{,}7]$
16	$f(x) = \ln(x) + 0.9\cos(x) + 1.7x$	$x \in [0{,}1; 10{,}1]$
17	$f(x) = \ln(x)\sin(2x) + 1.8x$	$x \in [0{,}5; 8]$
18	$f(x) = \ln(0.86x)\cos(0.75x) + 1.9x$	$x \in [1{,}5; 6{,}1]$
19	$f(x) = 1/x + \sin(0.56x) + 2x$	$x \in [0{,}4; 5{,}5]$
20	$f(x) = 1.23/x + \cos(1.44x) + 2.1x$	$x \in [0{,}3; 4{,}7]$

№	Функція	Інтервал побудови
21	$f(x) = 1/x * \sin(x) + 2.2x$	$x \in [0{,}5; 5]$
22	$f(x) = 1/x * \cos(3.1x) + 2.3x$	$x \in [1{,}5; 6{,}1]$
23	$f(x) = \sqrt{x+4}/(x+3) * \cos(x) + 2.4x$	$x \in [0{,}4; 5{,}5]$
24	$f(x) = \sin(3x) + 2.5x$	$x \in [0{,}3; 4{,}7]$

2.2 Побудувати графік параметрично заданої функції, за допомогою команд plot та comet.

№	Функція
1	$x(t) = t - \sin(t);\ y(t) = 1 - \cos(t)$
2	$x(t) = 2\sin(t) - \frac{2}{3}\sin(2t);\ y(t) = 2\cos(t) - \frac{2}{3}\cos(2t)$
3	$x(t) = 9\sin(0.1t) - 0.5\sin(0.9t);\ y(t) = 9\cos(0.1t) - 0.5\cos(0.9t)$
4	$x(t) = \cos(t);\ y(t) = \sin(\sin(t))$
5	$x(t) = 1 - \cos(2t);\ y(t) = 1 - \sin(2t)$
6	$x(t) = e^{-2t}\sin(t);\ y(t) = \cos(t)$
7	$x(t) = e^{-t}\cos(t);\ y(t) = \sin(t)$
8	$x(t) = e^{-t}\cos(t);\ y(t) = e^{t}\sin(t)$
9	$x(t) = \sin(2t) + \cos^2(t);\ y(t) = \cos(t) + \sin^2(t)$
10	$x(t) = \sin(t - \pi);\ y(t) = \sin(\cos(t))$
11	$x(t) = t + \cos(2t);\ y(t) = t + \cos^2(t)$
12	$x(t) = \sin(t(t - 2\pi));\ y(t) = \sin(t)$
13	$x(t) = \sin(t(t - 2\pi));\ y(t) = \sin(t)\cos(t)$
14	$x(t) = \sin(t) + \cos^3(t);\ y(t) = \sin(t)\cos(t)$
15	$x(t) = \cos(2t);\ y(t) = \sin(2t - \pi)$

2.3 Візуалізувати різними способами графіки функції двох змінних на прямокутній області завдання:

- каркасною поверхнею;
- залитої кольором каркасною поверхнею;
- промаркірованими лініями рівня(самостійно вибрати значення функції, що відображаються лініями рівня);
- освітленою поверхнею.

Розташувати графіки в окремих графічних вікнах і в одному вікні з відповідним числом осей.

№	Функція	Інтервал побудови
1	$z(x, y) = \sin(x)e^{-3y}$	$x, y \in [-1, 1]$
2	$z(x, y) = \sin^2(x)\ln(y)$	$x, y \in [-1, 1]$

№	Функція	Інтервал побудови
3	$z(x,y)=\sin^2(x-2y)e^{-\lvert y\rvert}$	$x,y\in[-1,1]$
4	$z(x,y)=\dfrac{x^2y^2+2xy-3}{x^2+y^2+1}$	$x,y\in[-1,1]$
5	$z(x,y)=\dfrac{\sin(xy)}{x}$	$x,y\in[-1,1]$
6	$z(x,y)=(\sin(x^2)+\cos(y^2))^{xy}$	$x,y\in[-1,1]$
7	$z(x,y)=\cos^2(x)\ln(y)$	$x,y\in[10,20]$
8	$z(x,y)=x^3\sin(xy)$	$x,y\in[-1,1]$
9	$z(x,y)=(x^2-2)\cos(y^2)$	$x,y\in[-1,1]$
10	$z(x,y)=(1+xy)(3-x)(4-y)$	$x,y\in[-10,10]$
11	$z(x,y)=e^{-\lvert x\rvert}(x^5+y^4)\sin(xy)$	$x,y\in[-5,5]$
12	$z(x,y)=(y^2-3)\sin\left(\dfrac{x}{\lvert y\rvert+1}\right)$	$x,y\in[-\pi,\pi]$
13	$z(x,y)=(x^3+y)\cos(e^y)$	$x,y\in[-10,10]$
14	$z(x,y)=\cos^2(x)\ln(y^2)$	$x,y\in[10,20]$
15	$z(x,y)=\arctan(x+y)(\arccos(x)+\arcsin(y))$	$x,y\in[-1,1]$
16	$z(x,y)=\dfrac{\sin(xy)}{x}$	$x,y\in[-1,1]$
17	$z(x,y)=(1+xy)(3-x)(4-y)$	$x,y\in[-10,10]$
18	$z(x,y)=(y^2-3)\sin\left(\dfrac{x}{\lvert y\rvert+1}\right)$	$x,y\in[-\pi,\pi]$
19	$z(x,y)=\dfrac{x^2y^2+2xy-3}{x^2+y^2+1}$	$x,y\in[-1,1]$
20	$z(x,y)=x^3\sin(xy)$	$x,y\in[-1,1]$

2.3 Контрольні запитання

1.Яка графічна функція використовується для побудови графіка функції у лінійному масштабі?

2.Перелічіть спеціальні графічні функції для побудови графіків функції у логарифмічному масштабі.

3.Яка функція дозволяє розділити графічне вікно на декілька вікон та вивести у кожному з них графіки різних функцій?

6.За допомогою якої команди відбувається завдання ліній сітки?

7.Як побудувати графік функції двох змінних як каркасну поверхню?

8.Опишіть формати команд оформлення графіків: титульний напис, підписи осей, легенду.

3 ЛАБОРАТОРНА РОБОТА №3
ПОШУК КОРЕНІВ АЛГЕБРАЇЧНИХ РІВНЯНЬ

Мета роботи: дослідити можливості середовища MATLAB щодо методів розв'язання алгебраїчних рівнянь, дослідити особливості роботи відповідних інструментів MATLAB.

3.1 Теоретичні відомості

3.1.1 Пошук коренів рівняння однієї змінної

3.1.1.1 Знаходження коренів довільних нелінійних рівнянь

Розв'язування цієї задачі, як правило розбивається на два етапи.

На першому етапі знаходимо наближене значення кореня, на другому етапі проводимо ітераційне уточнення кореня.

На етапі знаходження наближеного значення кореня знаходимо достатньо вузькі відрізки (або відрізок, якщо корінь єдиний), які містять один і лише один корінь рівняння.

Функція *fzero* дозволяє приблизно обчислити корінь рівняння на *деякому інтервалі* або найближчому до *заданого початкового наближення*. У цій функції реалізований алгоритм, який є комбінацією відомого методу половинного ділення (бісекції, дихотомії), методу січних і методу зворотної квадратичної інтерполяції.

У простому випадку *fzero* викликається з двома вхідними і одним вихідним аргументам:

```
>>x=fzero(fun,x0)
```

де *x0* – початкове наближення до кореня, *x* – знайдене наближене значення кореня.

Аргумент *fun* може бути вказаний одним із способів:

- як формула в лапках;
- як ім'я inline-функції (без лапок);
- як ім'я m-файла (у лапках і без розширення m).

Аргумент *x0* може бути вказаний одним з двох способів:

- як вектор *[a,b], a<b*, що вказує на інтервал невизначеності (на цьому інтервалі функція змінює знак), всередині якого знаходиться корінь;
- як скалярне значення, в околиці якого розшукується корінь.

Наприклад, потрібно знайти рішення рівняння

$y(x) = \sin\left(\frac{1}{x}\right) - 0.25 = 0$. на відрізку [0,3; 10]

Перед знаходженням коренів корисно побудувати графік функції, що входить в ліву частину рівняння. Для цього скористаємося функцією *fplot*.

```
>> fplot('sin(1/x)-0.25',[0.3 10])
>> grid on
```

Як видно з графіка (рис.***), шуканий корінь знаходиться в околицях точки x=4.

Застосуємо варіант пошуку кореня, коли задається інтервал невизначеності (два значення незалежної змінної, у яких функція має різний знак. Це означає, що десь в середині цього інтервалу функція перетинає відмітку *y=0*, тобто там є корінь рівняння).

```
>> x=fzero('sin(1/x)-0.25',[2 6])
```

Тут *[2 6]* — інтервал невизначеності, так як *y(2)= 0.23*, а *y(6)= -0.084*. Отже, функція змінює знак.

Як результат отримуємо

```
x =
   3.957570802030866
```

Перевірити, чи дійсно це корінь рівняння, дуже просто — потрібно підставити отриманий результат у вихідне рівняння:

```
>> sin(1/3.957570802030866)-0.25

ans =

     0
```

Так як результат рівний нулю, то рішення знайдене вірно.

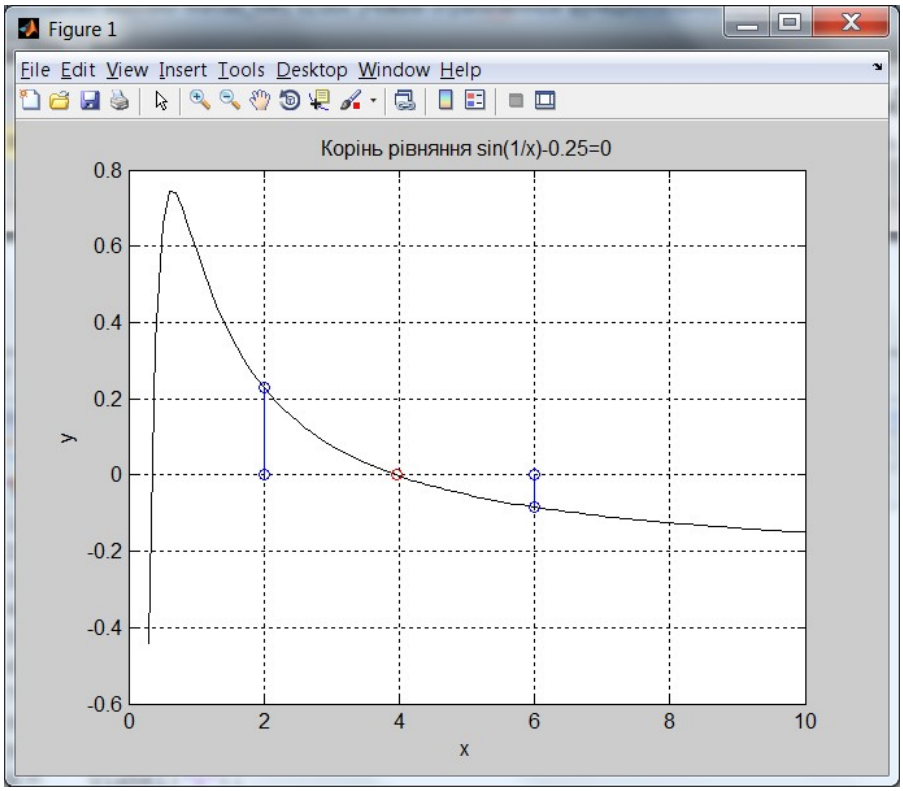

Рис. 3.1. Графічне рішення рівняння

Повторимо процедуру, відшукуючи корінь в околиці деякої точки:

```
>> x=fzero('sin(1/x)-0.25',5)

x =

   3.957570802030866
```

Якщо при пошуку кореня використати можливості inline-функцій, то код пошуку рішення матиме вигляд

```
>> equation=inline('sin(1/x)-0.25') % оголошення inline-функції для лівої частини рівняння (її назва, потім ключове слово inline та в дужках і одинарних лапках — власне вигляд лівої частини)
>> x=fzero(equation,5) % пошук кореня в околицях точки x=5

x =

   3.957570802030866
```

Як бачимо, отриманий результат ідентичний попереднім.

3.1.1.2 Розв'язування невеликих невироджених СЛАР

Для розв'язування невеликих не вироджених систем лінійних алгебраїчних рівнянь (СЛАР) можна використовувати оператор косої риски \ .

При цьому потрібно ввести два масиви: матрицю коефіцієнтів системи і вектор правої частини (вільних членів).

Наприклад, нехай потрібно рішення наступної системи:

$$\begin{cases} x_2 + 3x_3 + 2x_4 - x_5 = 11 \\ x_1 + 3x_2 + x_3 + 5x_4 = 2 \\ 3x_1 + 4x_2 + x_3 + 4x_4 + 2x_5 = 5 \\ 4x_1 + 2x_3 + x_4 + 3x_5 = 4 \\ 3x_1 + 2x_2 + x_3 + 4x_4 + 2x_5 = 5 \end{cases}$$

Введемо матрицю системи A і вектор вільних членів B.

```
>>A= [0 1 3 2 -1
     1 3 1 5 0
     3 4 1 4 2
     4 0 2 1 3
     3 2 1 4 2  ];
>>B=[11; 2; 5; 4; 5];
>>X=A\B
X=
-73.000000000000000
   0
  23.333333333333336
  10.333333333333334
  79.666666666666671
```

Аналогічний результат можна отримати, скориставшись властивістю зворотної матриці для такої системи:

$$A \cdot X = B \;\Rightarrow\; X = A^{-1}B$$

```
>> A1=inv(A);
>> X=A1*B
X =
 -73.000000000000000
                     0
  23.333333333333332
  10.333333333333334
  79.666666666666671
```

Перевірити розв’язок можна, підставивши знайдені корені в рівняння системи (це рівнозначно множенню матриці *A* на вектор *X*. Введемо вектор *B1* – результат підстановки.

```
>> B1=A*x
B1 =
11
2
5
4
5
```

Можемо переконатися, що значення вектора *B1 =B* початкової СЛАР:
B=[11; 2; 5; 4; 5];

Алгоритм розв’язку СЛАР визначається MATLAB виходячи із структури матриці коефіцієнтів системи. Якщо матриця є виродженою (її визначник = 0), то цей спосіб не підходить. Тому потрібно перевіряти визначник матриці на нерівність 0.

3.1.1.3 Розв’язування СЛАР за допомогою функції *linsolve*

Більші можливості надає функція *linsolve*, яка допускає вибір методу вирішення відповідно до властивостей матриці. Але вона також може використовуватися тільки для невироджених матриць.

В найпростішому випадку виклик *linsolve* має вигляд:

```
>> x=linsolve(A,B)
```

де *A* – матриця коефіцієнтів, *B* – вектор вільних членів.
Застосовуючи її до нашої СЛАР отримаємо розв’язок:

```
>> X=linsolve(A,B)
X =
 -73.000000000000000
                     0
  23.333333333333336
  10.333333333333334
  79.666666666666671
```

Для вибору методу розв’язування СЛАР потрібно сформувати управляючу структуру *options*, яка містить поля із значеннями true або false

залежно від властивостей матриці і вказати її третім вхідним аргументом функції *linsolve*.

```
>>options.SYM=true;
>>y=linsolve(A,b,options)
```

Параметри, які визначають властивості матриці, наведено в таблиці.

Таблиця 3.1. Прапорці, які визначають властивості матриці СЛАР

Прапорець	Властивість матриці
LT	Нижня трикутна
UT	Верхня трикутна
UHESS	Верхня майже трикутна
SYM	Симетрична
POSDEF	Позитивно визначена
RECT	Прямокутна

3.1.2 Пошук розв'язку системи нелінійних рівнянь

Формально задача пошуку розв'язку системи нелінійних рівнянь

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ ..... \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

може бути представлена в еквівалентній формі як задача пошуку кореня одного рівняння, $f(x) = (f(x)_1, f_2(x), ..., f_n(x))$ і $x = (x_1, x_2, ..., x_n)$. Команда *fsolve* призначена для розв'язання систем нелінійних рівнянь виду $f(x) = 0$, де *x* — вектор невідомих, а *f* — функція, значенням якої є вектор або матриця. Алгоритм роботи команди *fsolve* використовує початкове значення *x0* і базується на мінімізації суми квадратів складових функції $f(x) = (f(x)_1, f_2(x), ..., f_n(x))$ методами Гауса-Ньютона і Льовенберга-Марквардта.

У простому випадку структура звернення до команди *fsolve* має вигляд:

X = fsolve(F,X0).

Головним призначенням команди *fsolve* є розв'язок систем нелінійних рівнянь.

Приклад 1. Розв'язати систему рівнянь:

$$\begin{cases} x_1 = -x_2 + \sin(\pi x_1) \\ x_1 = x_2 + \cos(\pi x_2) \end{cases}$$

Випишемо m-файл *funsc*, значення якого сформуємо у вигляді вектора-стовпця:

```
function  y =  funsc(x)
y=[x(1)+x(2)-sin(pi*x(1)); x(1)-x(2)-cos(pi*x(1))];
```

Використовуємо команду *fsolve*. Зауважимо, що кожне звернення до цієї команди передбачає завдання різних початкових значень (*xl, x2*).

Оскільки команда *fsolve* повертає вектор-стовпець значень функцій в знайденій точці, то можна сформувати два вихідних параметра, щоб оцінити точність розв'язання. Звернемо увагу на те, що координати початкової точки теж представлені у вигляді вектора-стовпця:

```
>> [x, f] =  fsolve(@funsc,[0.2;1],optimset('Display', ' off'))
     x =
     0.5000
     0.5000
     f =
     1.0e-007 *
     0.2139
     -0.0000
```

3.1.3 Рішення окремих рівнянь та їх систем у символьному вигляді

MATLAB можна використовувати для рішення алгебраїчних і трансцендентних рівнянь і систем рівнянь, заданих у вигляді масиву символьних виразів. Основний інструмент для цього - функція *solve*. Ця функція входить до складу Symbolic Math Toolbox.

Вона може викликатися в різній формі:

solve (E1, E2, ..., EN)

solve (E1, E2, ..., EN, var1, var2, ..., varN)

Тут *E1, E2, ..., EN* — символьні вирази або змінні, в яких вони містяться, а *var1, ..., varN* — змінні, відносно яких слід вирішити систему рівнянь *E1=0, E2=0, ..., EN=0*. Зрозуміло, перша форма виклику цієї функції допустима лише у випадку, коли немає неоднозначностей щодо того, що саме слід знайти. Функція *solve* повертає єдиний символьний вираз, якщо рівняння (система рівнянь) має єдине рішення і вектор рішень в іншому випадку. Якщо рівняння

містить періодичні функції і може тому мати нескінченне число рішення, функція обмежується тим, що повертає корінь за один період в околиці нуля.

У процесі символьних обчислень використовуються змінні і константи особливого типу, так звані символьні об'єкти. Хоча зазвичай в коді MATLAB тип змінних визначається динамічно і немає потреби оголошувати його явно, для символьних об’єктів справа йде інакше. Для оголошення символьних змінних служить команда *syms*, яка в якості аргументів приймає імена змінних, перераховані через пробіл. Наприклад, так:

```
>> syms x y
>> syms ab real % оголошуються об’єкти позначають дійсні змінні
```

Оголошення символьних констант здійснюється за допомогою функції *sym*. Вона може приймати в якості аргументу рядок, що містить спеціальні змінні, чисельне вираження або виклик функції, як в прикладах нижче:

```
>> sym_pi = sym('pi')
>> sym_delta = sym('1/10')
>> sym_sqroot2 = sym('sqrt(2)')
```

Після оголошення символьні змінні можна використовувати приблизно так само, як і зі звичайні числові. Зокрема, для них визначені оператори + - * /, за допомогою яких можна складати символьні вирази:

```
>> syms s t A
>> f = s ^ 2 + 4 * s + 5
f =
s ^ 2 + 4 * s + 5

>> g = s + 2
g =
s + 2

>> h = f * g
h =
(s ^ 2 + 4 * s + 5) * (s + 2)

>> z = exp (-s * t)
z =
exp (-s * t)
```

```
>> y = A * exp (-s * t)
y =
A * exp (-s * t)
```

У наведених командах символьні змінні *s, t, A* використовуються для складання символьних виразів, створюючи нові символьні змінні *f, g, h, z, y*. При цьому останні автоматично оголошуються символьними і їх значення не обчислюється і ніяк не перетвориться.

Розглянемо приклади.

Єдине рішення

```
>> syms s;
>> E = s + 2;
>> S = solve (E);
s =
-2
```

Кілька рішень

```
>> syms s;
>> D = s ^ 2 + 6 * s + 9;
>> S = solve (D)
s =
[-3]
[-3]
```

Рівняння з параметрами

```
>> syms theta x z;
>> E = z * cos (theta) - x;
>> Theta = solve (E, theta)
theta =
acos (x / z)
```

Рівняння з комплексними коренями

```
>> syms x;
>> E = exp (2 * x) + 4 * exp (x) - 32;
>> X = solve (E)
x =
[log (-8)]
[log (4)]
>> log (-8)
ans =
2.0794 + 3.1416i
>> log (4)
ans =
1.3863
```

Рівняння з нескінченним числом рішень

```
>> E = cos (2 * theta) - sin (theta);
>> solve (E)
theta =
[-1 / 2 * pi]
[1/6 * pi]
[5/6 * pi]
```

Функція *solve* може використовуватися і для рішень системи рівнянь.

Рішення системи рівнянь

$$\begin{cases} x^2 - y = 2 \\ y - 2x = 5 \end{cases}$$

можна знайти так:

```
>> syms x y;
>> eq1=x^2-y-2;
>> eq2=y-2*x-5;
>> res=solve(eq1, eq2, 'x, y')
res =
    x: [2x1 sym]
    y: [2x1 sym]
```

Ця система рівнянь має два рішення. Програма MATLAB видає рішення, виводячи два значення *x* і два значення *y* для цих рішень. Таким чином, перше рішення складається з першого значення *x* і першого значення *y*. Зверніть увагу, що MATLAB не видав рішення автоматично. Замість цього на екрані з'явився напис, який говорить про знайдені дві пари коренів. Так сталося через те, що при вирішенні системи рівнянь функція повертає масив структур, де кожна структура відповідає кореню.

```
ans =
 2*2^(1/2) + 1
 1 - 2*2^(1/2)
>> res.y
ans =
  4*2^(1/2) + 7
 7 - 4*2^(1/2)
```

або, у вигляді чисел:

```
>> double(res.x)
ans =
    3.8284
   -1.8284
>> double(res.y)
```

```
ans =
   12.6569
    1.3431
```

Функція double обчислює значення символьного виразу.

Щоб побачити вектори значень *x* і *y*, введіть *res.x* і *res.y*. щоб побачити окремі значення, введіть *res.x(1)* і *res.y(1)*, і т.д.

Рішення цієї системи можна отримати відразу у вигляді векторів, застосувавши форму запису

```
>> [x y]=solve(eq1, eq2, 'x, y')
 x =
 2*2^(1/2) + 1
 1 - 2*2^(1/2)
  y =
  4*2^(1/2) + 7
 7 - 4*2^(1/2)
```

Побудувавши графіки рівнянь (рис. ***), що входять до складу системи, можна візуально перевірити правильність рішення. Для побудови графіки функцій у символьному вигляді використовуються такі ж функції, що й для звичайних графіків, тільки з додаванням префіксів ez- перед назвою. Наприклад, *ezplot, ezmesh, ezsurf* і.т.д.

```
>> ezplot(eq1,[-5 5 -5 15]);% будує графік першого рівняння
```

у межах *-5<x<5, -5<y<15*

```
>> hold on
>> ezplot(eq2,[-5 5 -5 15]);
>> grid on
>> xlabel('x');
>> ylabel('y');
>> title('Криві системи рівнянь');
```

Як бачимо, точки перетину графіків відповідають знайденим кореням рівняння.

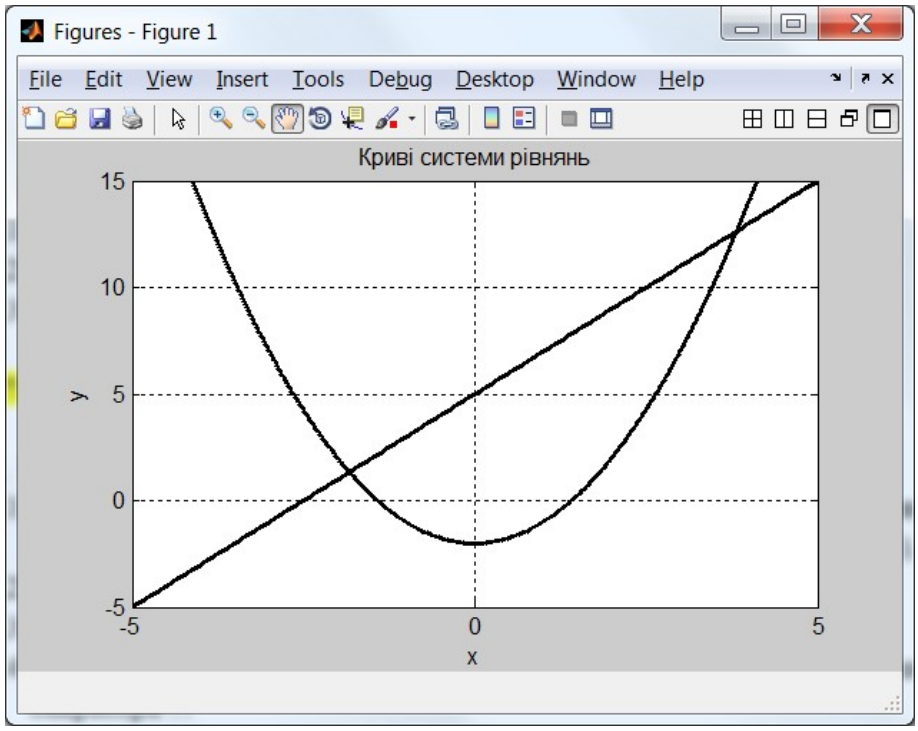

Рис. 3.2. Графіки функцій системи рівнянь

3.2 Порядок виконання роботи

3.1. Розв’язок рівняння однієї змінної.

3.1.1. Сформувати inline-функцію у відповідності до варіанта.

3.1.2. Оцінити інтервал невизначеності.

3.1.3. Знайти корінь (корені) рівняння у визначеному інтервалі.

№	Рівняння	№	Рівняння
1	$x^4 - 4x^2 + 12x - 8 = 0$	11	$x^2 - 2x + \ln(x) = 0$
2	$2^x + 2x^2 - 3 = 0$	12	$x^2 - 2\ln(x+2) = 0$
3	$x^3 + 2x - 13 = 0$	13	$x^2 + arctg(x) - 0{,}5 = 0$
4	$x \cdot e^{2x} - 4 = 0$	14	$ctg(0{,}8x) - 2x^2 = 0$
5	$x^5 + 5x + 1 = 0$	15	$x^5 + 18x^3 - 34 = 0$
6	$(x-2)^2 - e^x = 0$	16	$2x \cdot e^{x^2} - 5 = 0$
7	$x^3 + 2x^2 - 11 = 0$	17	$2e^{-x^2} - 3x + 4 = 0$
8	$x^2 - \cos(1{,}2x) - 1 = 0$	18	$2x - 3\sin(2x) - 1 = 0$
9	$2^x + 2x^2 - 5 = 0$	19	$x^2 - 2\ln(x+3) = 0$
10	$x \cdot e^{2x} - 9 = 0$	20	$x^3 + 2x - 19 = 0$

3.2. Рішення системи лінійних алгебраїчних рівнянь.

3.2.1. Для конкретного варіанту завдання вибрати систему лінійних рівнянь і побудувати матрицю її коефіцієнтів (A) і вектор вільних членів (B).

3.2.2. Ввести значення елементів матриці і вектора в MATLAB.

3.2.3. Знайти визначник матриці. Якщо він = **0**, то матриця вироджена.

3.2.4. Якщо визначник не рівний **0**, розв'язати систему з використанням оператора \ .

3.2.5. За зовнішнім виглядом матриці визначити її властивості (по таблиці 3.2). Якщо жодна з властивостей не виконується, розв'язати систему за допомогою функції linsolve з двома аргументами, інакше – третім аргументом вказати потрібну властивість за допомогою структури *options*.

3.2.6. Розв'язати систему методом зворотної матриці.

3.2.7. Порівняти результати, зробити висновки.

№	Система рівнянь	№	Система рівнянь
1	$\begin{cases} 2x_1 + x_2 + 2x_3 + 3x_4 = 8 \\ 3x_1 + 3x_3 = 6 \\ 2x_1 - x_2 + 3x_4 = 4 \\ x_1 + 2x_2 - x_3 + 2x_4 = 4 \end{cases}$	11	$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = -4 \\ x_1 - 3x_2 - 6x_4 = -7 \\ 2x_1 - x_3 + 2x_4 = 2 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = -2 \end{cases}$
2	$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 22 \\ 2x_1 + 3x_2 + x_3 + 2x_4 = 17 \\ x_1 + x_2 + x_3 - x_4 = 8 \\ x_1 - 2x_3 - 3x_4 = -7 \end{cases}$	12	$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 = 26 \\ 2x_1 + 3x_2 + 4x_3 + x_4 = 34 \\ 3x_1 + 4x_2 + x_3 + 2x_4 = 26 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 26 \end{cases}$
3	$\begin{cases} 9x_1 + 10x_2 - 7x_3 - x_4 = 23 \\ 7x_1 - x_3 - 5x_4 = 37 \\ 5x_1 - 2x_3 + x_4 = 22 \\ 4x_1 + x_2 + 2x_3 + 3x_4 = 26 \end{cases}$	13	$\begin{cases} 2x_1 - 8x_2 - 3x_3 - 2x_4 = -18 \\ x_1 - 2x_2 + 3x_3 - 2x_4 = 28 \\ x_2 + x_3 + x_4 = 10 \\ 11x_2 + x_3 + 2x_4 = 21 \end{cases}$
4	$\begin{cases} 6x_1 - x_2 + 10x_3 - x_4 = 158 \\ 2x_1 + x_2 + 10x_3 + 7x_4 = 128 \\ 3x_1 - 2x_2 - 2x_3 - x_4 = 7 \\ x_1 - 12x_3 + 2x_3 - x_4 = 17 \end{cases}$	14	$\begin{cases} 2x_1 - x_2 + 4x_3 + x_4 = 66 \\ 2x_2 - 6x_3 + x_4 = -63 \\ 8x_1 - 3x_2 + 6x_3 - 5x_4 = 146 \\ 2x_1 - 7x_2 + 6x_3 - x_4 = 80 \end{cases}$
5	$\begin{cases} 2x_1 + x_2 + 2x_3 + 3x_4 = 8 \\ 3x_1 + 3x_3 = 6 \\ 2x_1 - x_2 + 3x_4 = 4 \\ x_1 + 2x_2 - x_3 + 2x_4 = 4 \end{cases}$	15	$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = -4 \\ x_1 - 3x_2 - 6x_4 = -7 \\ 2x_1 - x_3 + 2x_4 = 2 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = -2 \end{cases}$

№	Система рівнянь	№	Система рівнянь
6	$\begin{cases} x_1+2x_2+3x_3+4x_4=22 \\ 2x_1+3x_2+x_3+2x_4=17 \\ x_1+x_2+x_3-x_4=8 \\ x_1-2x_3-3x_4=-7 \end{cases}$	16	$\begin{cases} x_1+2x_2+3x_3+4x_4=26 \\ 2x_1+3x_2+4x_3+x_4=34 \\ 3x_1+4x_2+x_3+2x_4=26 \\ 4x_1+x_2+2x_3+3x_4=26 \end{cases}$
7	$\begin{cases} 9x_1+10x_2-7x_3-x_4=23 \\ 7x_1-x_3-5x_4=37 \\ 5x_1-2x_3+x_4=22 \\ 4x_1+x_2+2x_3+3x_4=26 \end{cases}$	17	$\begin{cases} 2x_1-8x_2-3x_3-2x_4=-18 \\ x_1-2x_2+3x_3-2x_4=28 \\ x_2+x_3+x_4=10 \\ 11x_2+x_3+2x_4=21 \end{cases}$
8	$\begin{cases} 6x_1-x_2+10x_3-x_4=158 \\ 2x_1+x_2+10x_3+7x_4=128 \\ 3x_1-2x_2-2x_3-x_4=7 \\ x_1-12x_3+2x_3-x_4=17 \end{cases}$	18	$\begin{cases} 2x_1-x_2+4x_3+x_4=66 \\ 2x_2-6x_3+x_4=-63 \\ 8x_1-3x_2+6x_3-5x_4=146 \\ 2x_1-7x_2+6x_3-x_4=80 \end{cases}$
9	$\begin{cases} x_1+2x_2+3x_3+4x_4=22 \\ 2x_1+3x_2+x_3+2x_4=17 \\ x_1+x_2+x_3-x_4=8 \\ x_1-2x_3-3x_4=-7 \end{cases}$	19	$\begin{cases} x_1+2x_2+3x_3+4x_4=26 \\ 2x_1+3x_2+4x_3+x_4=34 \\ 3x_1+4x_2+x_3+2x_4=26 \\ 4x_1+x_2+2x_3+3x_4=26 \end{cases}$
10	$\begin{cases} 9x_1+10x_2-7x_3-x_4=23 \\ 7x_1-x_3-5x_4=37 \\ 5x_1-2x_3+x_4=22 \\ 4x_1+x_2+2x_3+3x_4=26 \end{cases}$	20	$\begin{cases} 2x_1-8x_2-3x_3-2x_4=-18 \\ x_1-2x_2+3x_3-2x_4=28 \\ x_2+x_3+x_4=10 \\ 11x_2+x_3+2x_4=21 \end{cases}$

3.3. Рішення системи нелінійних алгебраїчних рівнянь.

3.3.1. За допомогою функції solve знайти точки перетину плоских кривих (див. таблицю).

3.3.2. Навести графічне рішення системи рівнянь (використати функцію ezplot).

№	Функції плоских кривих
1	$\frac{x^2}{4}+\frac{(y-1)^2}{25}=1,\ \ \frac{y}{2}-\frac{x}{4}=1$
2	$(x-1)^2+(y-1)^2=4,\ \ y=2x$
3	$x^2+(y-5)^2=25,\ \ y=x^2$
4	$\frac{x^2}{36}+\frac{y^2}{16}=1,\ \ (x+4)^2+y^2=9$
5	$\frac{x^2}{9}+\frac{y^2}{4}=1,\ \ 4x-12y=0$
6	$y=x^{-1},\ \ x^2+y^2=9$

№	Функції плоских кривих
7	$\frac{(x+1)^2}{25}+\frac{(y-2)^2}{9}=1,\ y=\frac{x^2}{6}$
8	$y=\frac{1}{x},\ \frac{x^2}{9}-\frac{y^2}{4}=1$
9	$(x+1)^2+y^2=16,\ y=\frac{x^2}{x-1}$
10	$x^2/36+y^2/4=1,\ y=6x-3x^3$
11	$\frac{(x+2)^2}{25}+\frac{(y-3)^2}{36}=1,\ y=\frac{x^2}{x-1}$
12	$x^2/25+(y+2)^2/16=1,\ y=3-x^2$
13	$y=x^2+3;\ 5y+2x-3=0$
14	$0.5x+0.25y=1,\quad y=2x^2-6x$
15	$\frac{x^2}{4}+\frac{(y-1)^2}{25}=1,\quad \frac{y}{2}-\frac{x}{4}=1$
16	$y=2x,\quad (x-1)^2+(y-1)^2=4$
17	$\frac{x^2}{9}-\frac{y^2}{4}=1,\quad y=\frac{1}{x}$
18	$2x-6y=0,\quad \frac{x^2}{9}+\frac{y^2}{4}=1$
19	$x^2+y^2=9,\ y=\frac{1}{x}$
20	$y=3.75x,\quad (x-1)^2+(y-1)^2=4$

3.3 Контрольні запитання

1. Рішення системи лінійних алгебраїчних рівнянь.
2. На які етапи поділяється процес розв'язування нелінійного рівняння?
3. Як розв'язується нелінійне рівняння або система нелінійних рівнянь у символьному вигляді в системі MATLAB?
5. Які методи розв'язування нелінійних рівнянь у системі MATLAB вам відомі?
6. З яких етапів складається задача знаходження нулів функції у системі MATLAB?
7. Які функції для розв'язування нелінійного рівняння в системі MATLAB ви знаєте?
8. Назвіть функції для розв'язування системи нелінійних рівнянь в MATLAB і особливості їх застосування.

4 ЛАБОРАТОРНА РОБОТА №4
ІНТЕРПОЛЯЦІЯ ТА АПРОКСИМАЦІЯ

Мета роботи: отримати навички роботи з файлами у середовищі Matlab, ознайомитись з можливостями та особливостями взаємодії з файлами.

4.1 Теоретичні відомості

4.1.1 Загальні відомості

Інтерполяція — в обчислювальній математиці спосіб знаходження проміжних значень величини по наявному дискретному наборі відомих значень. Багатьом з тих, хто стикається з науковими та інженерними розрахунками часто доводиться оперувати наборами значень, отриманих дослідним шляхом або методом випадкової вибірки. Як правило, на підставі цих наборів потрібно побудувати функцію, на графік якої могли б з високою точністю потрапляти інші одержувані значення. Така задача називається апроксимацією. Інтерполяцією називають такий різновид апроксимації, при якій крива побудованої функції проходить точно через наявні точки даних.

Існують різні методи побудови інтерполяційного полінома по заданих точках. Їх основна ціль – побудувати таку функцію (поліном) *g(x),* значення якої не тільки співпадають із значеннями y_k в заданих точках, але і має найменше відхилення від *f(x)* в інших точках відрізка [*a,b*].

4.1.2 Інтерполяція поліномами за методом найменших квадратів

Табличні дані дуже зручно інтерпретувати як певну функцію, зокрема поліном. Звідси виникає задача про побудову поліноміальної функції для наближення певних вихідних даних. Стандартний метод інтерполяції, реалізований в MATLAB – наближення поліномом по методу найменших квадратів.

Побудова поліномів фіксованого ступеня для наближення табличної функції однієї змінної робиться за допомогою функції *polyfit* з трьома вхідними параметрами:

p=polyfit(x,y,n)

де *x* — вектор значень абсцис;
y — вектор значень ординат;
n — ступінь полінома.
Функція polyfit дозволяє знаходити коефіцієнти полінома (вектор *p*).

Наприклад, маємо таблично задану функцію $y = f(x)$ з таким набором вихідних даних

Таблиця 4.1. Таблично задана функція

x_i	-0,3	0,2	0,7	1,1	1,5	1,6	2,0
y_i	-2,5	-3,8	-2,1	0,2	1,5	2,3	2,7

Дослідимо роботу методу інтерполяції, використавши поліноми різної степені (від 2 до 6).

Для цього:

1. інтерполюємо табличні дані, щоб знайти коефіцієнти поліномів відповідної степені. Використовуємо функцію polyfit;
2. щоб побудувати графіки отриманих поліномів, по-перше, створюємо вектор аргумента *xj* на відрізку $[a; b]$ з достатньо малим кроком і, по-друге, передаємо його як аргумент функції *polyval(p,xj)*, яка обчислює значення полінома з коефіцієнтами *p* у точках *xj*;
3. будуємо графіки вихідної табличної функції та отриманих поліномів для візуального аналізу.

```
%вихідні табличні дані
x=[-0.3 0.2 0.7 1.1 1.5 1.6 2.0];
y=[-2.5 -3.8 -2.1 0.2 1.5 2.3 2.7];
%пошук коефіцієнтів поліномів різного степеню
p2=polyfit(x,y,2);
p3=polyfit(x,y,3);
p4=polyfit(x,y,4);
p5=polyfit(x,y,5);
p6=polyfit(x,y,6);
% побудова вектора аргументів на відрізку, заданому табличною
функцією
xj=-0.3:0.01:2.0;
%формування вектору значень поліномів у точках xj
P2=polyval(p2,xj);
P3=polyval(p3,xj);
P4=polyval(p4,xj);
P5=polyval(p5,xj);
P6=polyval(p6,xj);
% побудова графіків
plot (x,y,'ko');
grid on
hold on
plot (xj,P2,'r-.', xj,P3,'g--', xj,P4,'c:', xj,P5,'b-', xj,P6,'k-
');
%оформлення графіків
xlabel('x');
ylabel('y');
```

```
title('Інтерполяція поліноміальними залежностями');
legend('табличні значення','n=2','n=3','n=4','n=5','n=6');
```

У результаті отримаємо графічне вікно, де містяться побудовані графіки

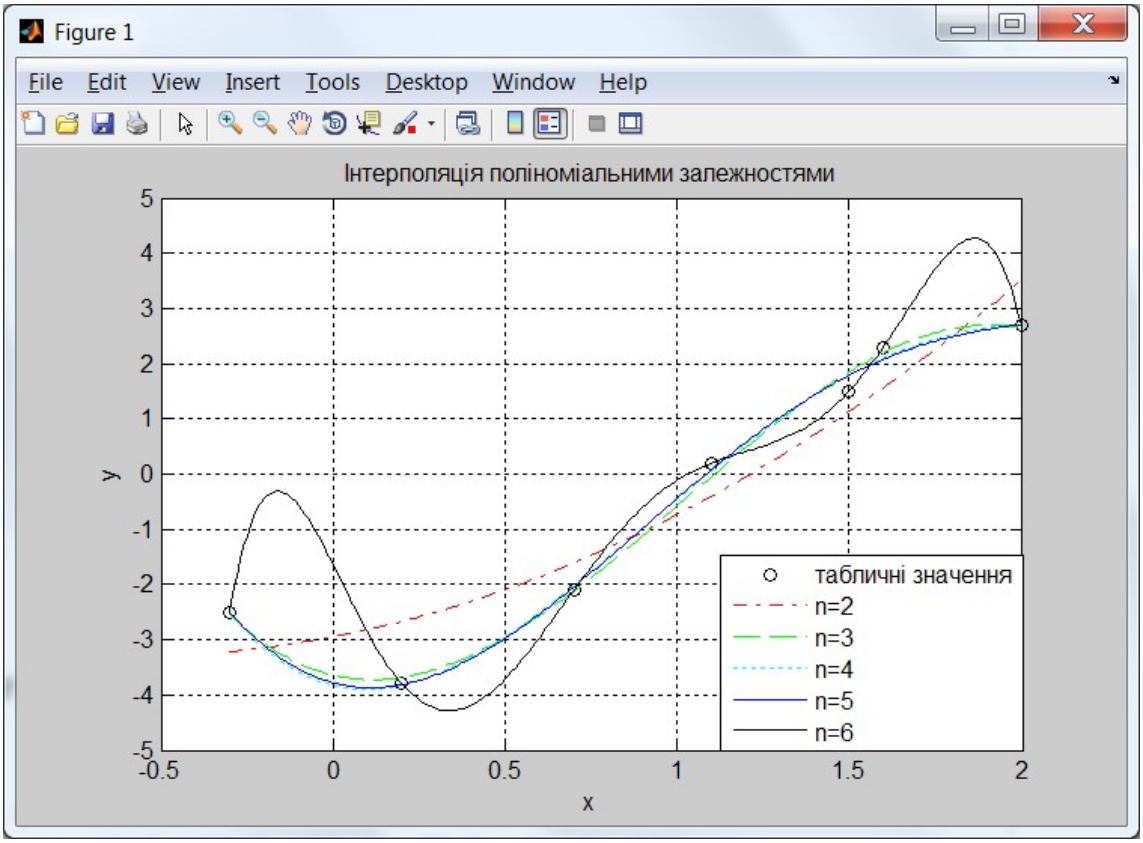

Рис. 4.1. Інтерполяція поліномами різного порядку

Слід зазначити, що наближення методом найменших квадратів не завжди дає пристойний результат. Крім цього, збільшення порядку полінома може призвести до погіршення точності пошуку значень у проміжних точках.

Обчислимо тепер значення знайдених поліномів у заданих точках *xi* та порівняємо з заданими *yi*:

```
>> Resi(1,:)=y;
>> Resi(2,:)=polyval(p2,x);
>> Resi(3,:)=polyval(p3,x);
>> Resi(4,:)=polyval(p4,x);
>> Resi(5,:)=polyval(p5,x);
>> Resi(6,:)=polyval(p6,x);
>> Resi
```

Результуюча матриця *Resi* містить у першій строці точні значення функції *yi* (вихідні), а в наступних — значення поліноміальних функцій, обчислених у точках *xi*.

```
Resi =
 -2.5000   -3.8000   -2.1000    0.2000    1.5000    2.3000    2.7000
 -3.2164   -2.6686   -1.6132   -0.4036    1.1309    1.5652    3.5056
 -2.5406   -3.6830   -2.1215   -0.0438    1.8473    2.1910    2.6505
 -2.4963   -3.8239   -2.0204    0.0597    1.7818    2.0910    2.7081
 -2.4981   -3.8145   -2.0430    0.0823    1.7852    2.0770    2.7110
 -2.5000   -3.8000   -2.1000    0.2000    1.5000    2.3000    2.7000
```

Як видно, з ростом порядку поліному обчислені у ключових точках значення наближаються до заданих, хоча можливі певні відхилення. Тому кінцевий результат інтерполяції повинен проходити перевірку дослідником.

4.1.3 Інтерполяція сплайнами

Сплайн — неперервна, гладка функція, що на відрізках області визначення дорівнює поліномам визначеної степені.

Найпростішим способом інтерполяції є наближення даних сплайном нульового порядку (на кожній ділянці ступінь полінома дорівнює нулю), при якому значення в кожній проміжній точці приймається рівним найближчому значенню, заданому в таблиці. В результаті дані наближаються східчастою функцією, а саме наближення називається інтерполяцією по сусідніх точках. Лінійна інтерполяція заснована на з'єднанні сусідніх точок відрізками прямих – табличні дані наближаються ламаною лінією. Для отримання більш гладкої функції слід застосовувати інтерполяцію кубічними сплайнами. Всі ці способи реалізовані у функції MATLAB *interp1*, у якості вхідних аргументів якої указуються координати абсцис (*x*) і ординат (*y*) табличної функції, координати абсцис проміжних точок, в яких обчислюються значення полінома і спосіб інтерполяції (*method*):

yi=interp1(x, y, xi, method)

Текстові константи, які задають спосіб інтерполяції (значення аргумента *method*):

'*nearest*' – наближення по сусідніх елементах;

'*linear*' – лінійна інтерполяція;

'*spline*' – інтерполяція кубічними сплайнами.

Вихідним аргументом *interp1* є вектор значень полінома в проміжних точках (*yi*).

Продемонструємо всі ці способи інтерполяції на вже відомому прикладі.

```
%Вихідні дані
x=[-0.3 0.2 0.7 1.1 1.5 1.6 2.0];
y=[-2.5 -3.8 -2.1 0.2 1.5 2.3 2.7];
%проміжні точки
xj=-0.3:0.01:2.0;
% обчислення східчастої функції в проміжних точках
ynearest=interp1(x,y,xj,'nearest')
% обчислення кусково-лінійної функції в проміжних точках
ylinear=interp1(x,y,xj,'linear')
% обчислення кубічного сплайна в проміжних точках
yspline=interp1(x,y,xj,'spline')
%подудова та оформлення графіків
plot (x,y,'ko');
grid on
hold on
plot(xj,ynearest,'k--',xj,ylinear,'k-.',xj,yspline,'k-');
xlabel('x');
ylabel('y');
title('Інтерполяція сплайнами');
legend('табличні          значення','по          сусідніх
точках','лінійна','кубічна');
```

У результаті отримаємо наступні графіки:

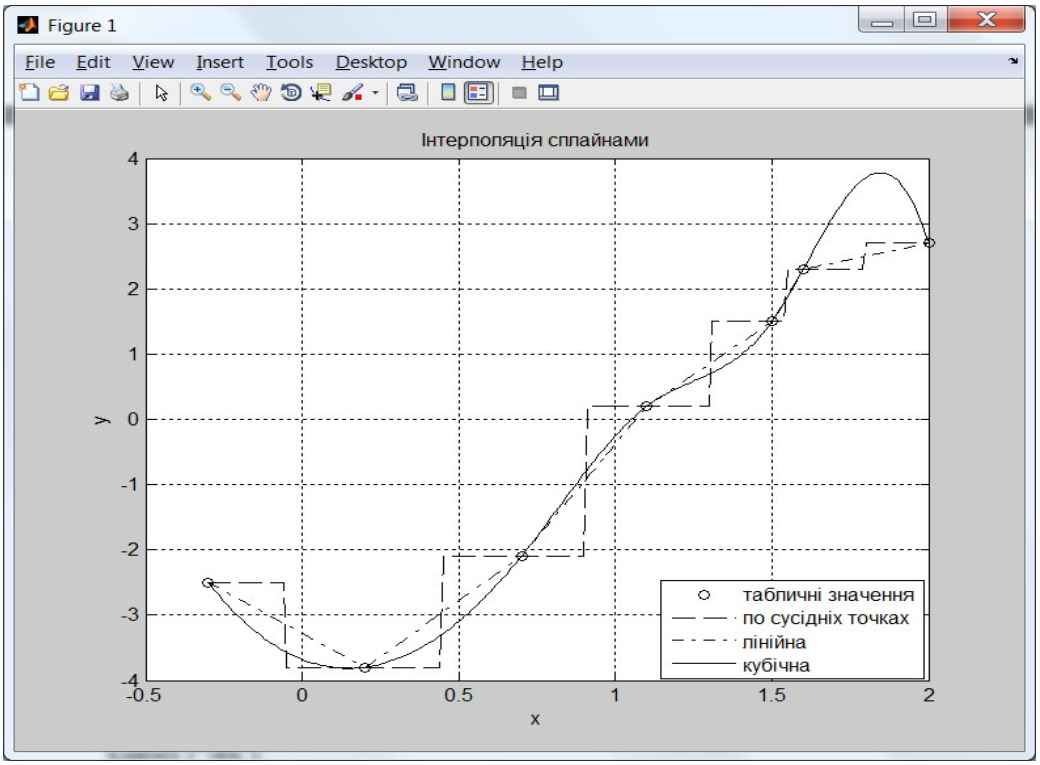

Рис. 4.2. Способи інтерполяції сплайнами

4.1.4 Інтерполяція двомірних та багатомірних даних

Функція *interp2* здійснює двовимірну інтерполяцію - важливу операцію при обробці зображень і графічного представлення даних. У найбільш загальній формі ця команда має вигляд

Zi = interp2 (X, Y, Z, Xi, Yi, method)

де *Z* є прямокутний масив, що містить значення двовимірної функції; *X* та *Y* являються масивами однакових розмірів, що містять точки, в яких задані значення двовимірної функції; *Xi* та *Yi* — матриці, що містять точки інтерполяції (тобто проміжнні точки, в яких потрібно обчислити значення функції); *method* - рядок, який визначає метод інтерполяції. У разі двовимірної інтерполяції можливі три різних методи:

- ступенева інтерполяція (*method = 'nearest'*). Цей метод дає кусково-постійну поверхню на області значень. Значення функції в інтерпольованій точці дорівнює значенню функції в найближчій заданій точці.
- білінійна інтерполяція (*method = 'bilinear'*). Метод забезпечує апроксимацію даних за допомогою білінійної поверхні (площини) на множині заданих значень двомірної функції. Значення в точці інтерполяції є комбінацією значень чотирьох найближчих точок. Цей метод швидше і вимагає менше пам'яті, ніж бікубічна інтерполяція.

- бікубічна інтерполяція (method = 'bicubic'). Даний метод апроксимує поверхню за допомогою бікубічних сплайнів. Значення в точці інтерполяції є комбінацією значень у шістнадцяти найближчих точках. Метод забезпечує значно більш гладку поверхню у порівнянні з білінійної інтерполяцією. Це може бути ключовим перевагою в додатках типу обробки зображень. Особливо ефективним даний метод є в ситуаціях, коли потрібна неперервність як інтерполюючих даних, так і їх похідних.

Всі ці методи вимагають, щоб *X* і *Y* були монотонними, тобто або завжди зростаючими або завжди спадаючими від точки до точки. Ці матриці слід сформувати, використовуючи функцію *meshgrid*. Якщо це неможливо, то потрібно переконатися, що «схема» точок імітує сітку, отриману функцією meshgrid. Якщо *X* і *Y* розподілені рівномірно, то можна прискорити обчислення, додаючи зірочку до рядка методу, наприклад, '**bicubic*'.

Порівняння методів інтерполяції

Наведений нижче приклад порівнює методи двовимірної інтерполяції у разі матриці даних розміру 6x6. Для прикладу сформуємо функцію з густою сіткою, а потім побудуємо інший масив з великим кроком. Цей масив даних використаємо як опорний для тестування методів інтерполяції. Задача методів — відтворити у проміжних точках поведінку початкової функції. Результати відобразимо на графіках.

```
%Вихідна функція
[X, Y]=meshgrid(0:0.02:1);
Z=sin(3*pi*X).*sin(3*pi*Y).*exp(-X.^2-Y.^2);
subplot(3,2,1)
surf(X,Y,Z)
shading interp
title('Вихідна функція')
%Таблично задана функція, яку інтерполюватимемо
[X, Y]=meshgrid(0:0.2:1);
Z=sin(3*pi*X).*sin(3*pi*Y).*exp(-X.^2-Y.^2);
subplot(3,2,2)
surf(X,Y,Z)
title('Таблично задана функція')
%Генераці сітки проміжних значень
[Xi, Yi]=meshgrid(0:0.02:1);
%інтерполяція по сусіднім значенням
ZiNear=interp2(X,Y,Z,Xi,Yi,'nearest');
subplot(3,2,3)
surf(Xi,Yi,ZiNear)
title('Інтерполяція сусідами')
```

```
%білінійна інтерполяція
ZiBilinear=interp2(X,Y,Z,Xi,Yi,'bilinear');
subplot(3,2,4)
surf(Xi,Yi,ZiBilinear)
title('Білінійна інтерполяція')
%бікубічна інтерполяція
ZiBicubic=interp2(X,Y,Z,Xi,Yi,'bicubic');
subplot(3,2,5)
surf(Xi,Yi,ZiBicubic)
title('Бікубічна інтерполяція')
ZiBicubic2=interp2(X,Y,Z,Xi,Yi,'*bicubic');
subplot(3,2,6)
surf(Xi,Yi,ZiBicubic2)
title('Швидка бікубічна інтерполяція')
```

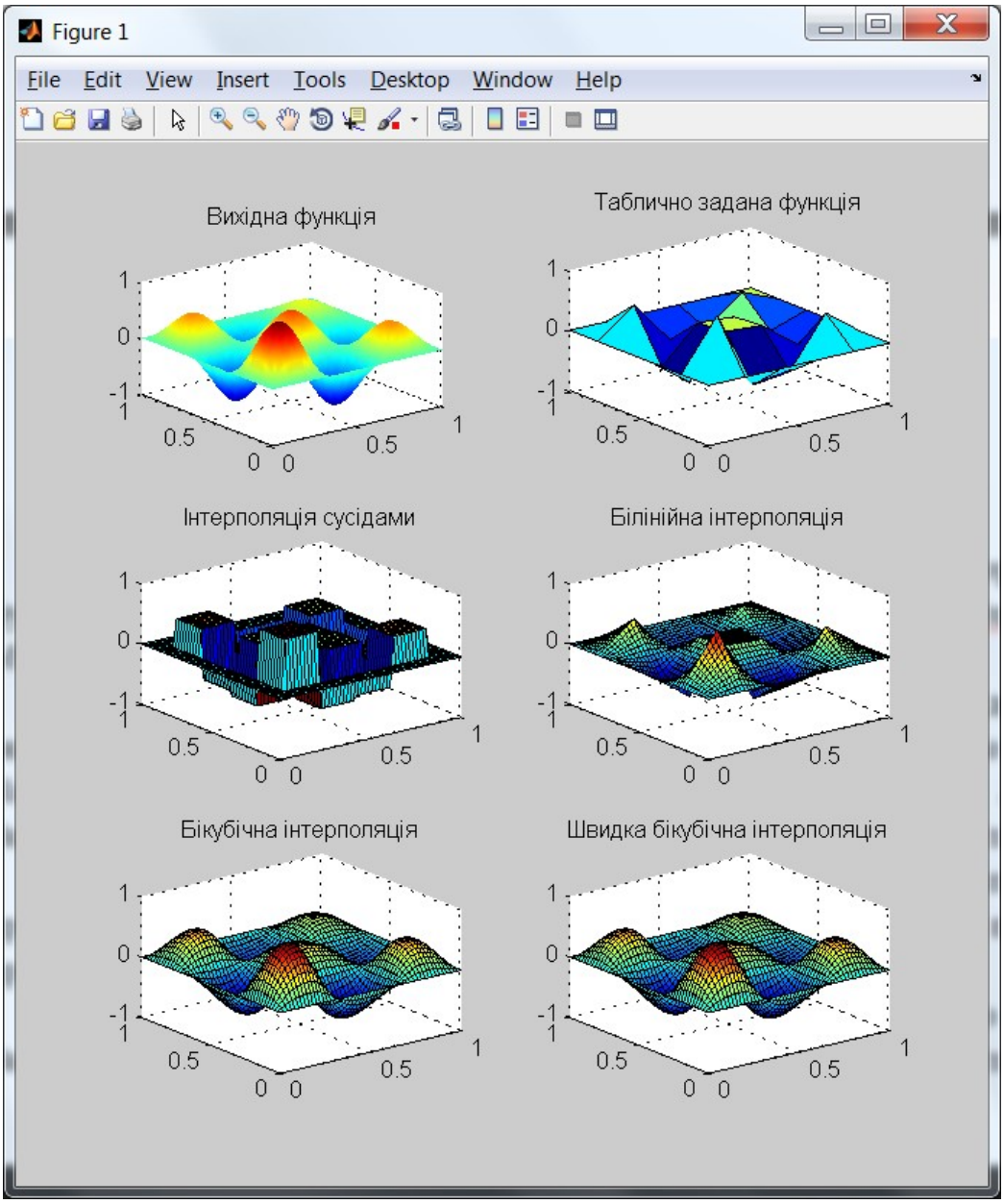

Рис. 4.3. Інтерполяція двомірних даних

Для інтерполяції тривимірних даних служить функція *interp3*, а для багатовимірних — *interpn*. Створення *n*-мірних сіток виконується функцією *ndgrid*. Процедура інтерполювання аналогічна такій для двовимірної задачі.

4.2 Порядок виконання роботи

5.1. Для заданої функції на інтервалі *[a; b]* згідно варіанта N сформувати табличну функцію у вигляді

$y_i = f(x_i), \quad x_i = 0.1 \cdot (b-a) \cdot i + a, \; i = 0...10$

N	Функція	Інтервал
1	$f(x) = x^{0.5} + \sin(x) + 0.1x$	$x \in [0{,}1; 10{,}1]$

N	Функція	Інтервал
2	$f(x)=\sqrt{x}+\cos(x)+0.2x$	$x\in[0{,}5;\,8]$
3	$f(x)=\sin(x)+\cos(2x)+0.3x$	$x\in[-1{,}5;\,9{,}1]$
4	$f(x)=\cos(x)+\sin(2x)+0.4x$	$x\in[-2{,}1;\,7{,}5]$
5	$f(x)=\sin(x)+0.5\cos(2x)+0.5x$	$x\in[0{,}3;\,8{,}7]$
6	$f(x)=\cos(x)+0.5\sin(2x)+0.6x$	$x\in[0{,}1;\,10{,}1]$
7	$f(x)=e^{-0.3x}\sin(x)+0.7x$	$x\in[0{,}5;\,8]$
8	$f(x)=e^{-0.3x}\cos(x)+0.8x$	$x\in[-1{,}5;\,9{,}1]$
9	$f(x)=x^2\cos(2x)+0.9x$	$x\in[-2{,}1;\,7{,}5]$
10	$f(x)=\sqrt{x}+\sin(3x)+x$	$x\in[0{,}3;\,8{,}7]$
11	$f(x)=\sqrt{x}+\cos(1.8x)+1.1x$	$x\in[0{,}1;\,10{,}1]$
12	$f(x)=x\cdot e^{-x}\sin(0.8x)+1.3x$	$x\in[0{,}5;\,8]$
13	$f(x)=e^{-\sin(x)}+1.4x$	$x\in[-1{,}5;\,9{,}1]$
14	$f(x)=e^{-\cos(x)}+1.5x$	$x\in[-2{,}1;\,7{,}5]$
15	$f(x)=\ln(x)+1.1\sin(x)+1.6x$	$x\in[0{,}3;\,8{,}7]$
16	$f(x)=\ln(x)+0.9\cos(x)+1.7x$	$x\in[0{,}1;\,10{,}1]$
17	$f(x)=\ln(x)\sin(2x)+1.8x$	$x\in[0{,}5;\,8]$
18	$f(x)=\ln(0.86x)\cos(0.75x)+1.9x$	$x\in[1{,}5;\,6{,}1]$
19	$f(x)=1/x+\sin(0.56x)+2x$	$x\in[0{,}4;\,5{,}5]$
20	$f(x)=1.23/x+\cos(1.44x)+2.1x$	$x\in[0{,}3;\,4{,}7]$

5.2. Для отриманої табличної функції побудувати поліноми 3…7-го ступенів з використанням функцій polyfit і polyval, графіки табличної функції і інтерполянта, обчислити значення поліномів в заданих точках *x1, x2, x3*.

$x_1=(b-a)/3+a,\quad x_2=0.5(a+b),\quad x_3=0.9(b-a)+a$

Порівняти отримані значення з точними.

5.3. Для отриманої табличної функції виконати інтерполяцію сплайнами різного виду. Проаналізувати результат.

4.3 Контрольні запитання

1. Визначення та задачі інтерполяції.
2. Інтерполяція методом найменших квадратів.
3. Функція інтерполяції методом найменших квадратів. Аргументи та використання.
4. Види сплайнів, їх особливості.

5. Інтерполяція сплайнами.
6. Двовимірна інтерполяція. Процедура та особливості.
7. Функції MATLAB для двовимірної інтерполяції.
8. Багатовимірна інтерполяція.

5 ЛАБОРАТОРНА РОБОТА №5
ПОШУК ЕКСТРЕМУМУ ФУНКЦІЇ

Мета роботи: дослідити методи пошуку екстремуму функції в середовищі MATLAB, отримати навички прикладного використання методів.

5.1 Теоретичні відомості

5.1.1 Мінімізація функції однієї змінної

Одна з важливих практичних задач - пошук мінімуму функції *f(x)* в деякому інтервалі зміни *x* - від *x1* до *x2*. Якщо потрібно знайти максимум такої функції, то достатньо поставити знак «мінус» перед функцією. Для вирішення цього завдання використовується функція *fminbnd*. ЇЇ синтаксис у загальному вигляді:

[X, fval, exitflag, output] = fminbnd (fun, x1, x2, options, p1, p2, ...)

Викликати функцію, залежно від обставин, можна у кількох варіантах:

- *fminbnd (fun, x1, x2)* повертає значення *x*, що є локальним мінімумом функції *fun (x)* на інтервалі *x1 <x <x2*.

- *fminbnd (fun, x1, x2, options)* подібна до описаної вище формою функції, але використовує параметри *tolX, maxfuneval, maxiter, display* з вектора *options,* попередньо встановлені за допомогою команди *optimset*.

- *fminbnd (fun, x1, x2, options, P1, P2, ...)* схожа з описаною вище, але передає в цільову функцію додаткові аргументи: *P1, P2*, ... Якщо потрібно використовувати параметри обчислень за замовчуванням, то замість *options* перед *P1, P2* необхідно ввести [] (порожній масив).

- *[x, fval] = fminbnd (...)* додатково повертає значення цільової функції *fval* в точці мінімуму.

- *[x, fval, exitflag] = fminbnd (...)* додатково повертає параметр *exitflag*, рівний 1, якщо функція зійшлася з використанням *options.tolX*, і 0, якщо досягнуто максимальне число ітерацій *options.maxiter*.

Тут використані наступні позначення: *x1, x2* - інтервал, на якому шукається мінімум функції; *P1, P2* ... – додаткові аргументи, крім *x*, що передаються у функцію; *fun* - рядок, що містить назву функції, яка буде мінімізована; *options* - вектор параметрів обчислень. В залежності від форми завдання функції *fminbnd* обчислення мінімуму виконується методами золотого перетину або параболічної інтерполяції.

Знайдемо, наприклад, мінімум функції

$f(x) = 3 \cdot \cos(x) \cdot e^{-0.25x}$

на відрізку $x \in [0; 8]$.

Завдання анонімної функції

```
>> f=@(x)3*cos(x).*exp(-0.25*x);
```

Завдання параметрів пошуку мінімуму

```
>> Options = optimset ('tolX', 1.e-10);
```

Пошук мінімуму:

```
>> [x, y] = fminbnd (f, 0,8, options)
x =
    2.8966
y =
   -1.4108
```

Правильність рішення можна перевірити, побудувавши графік функції на заданому проміжку (рис.)

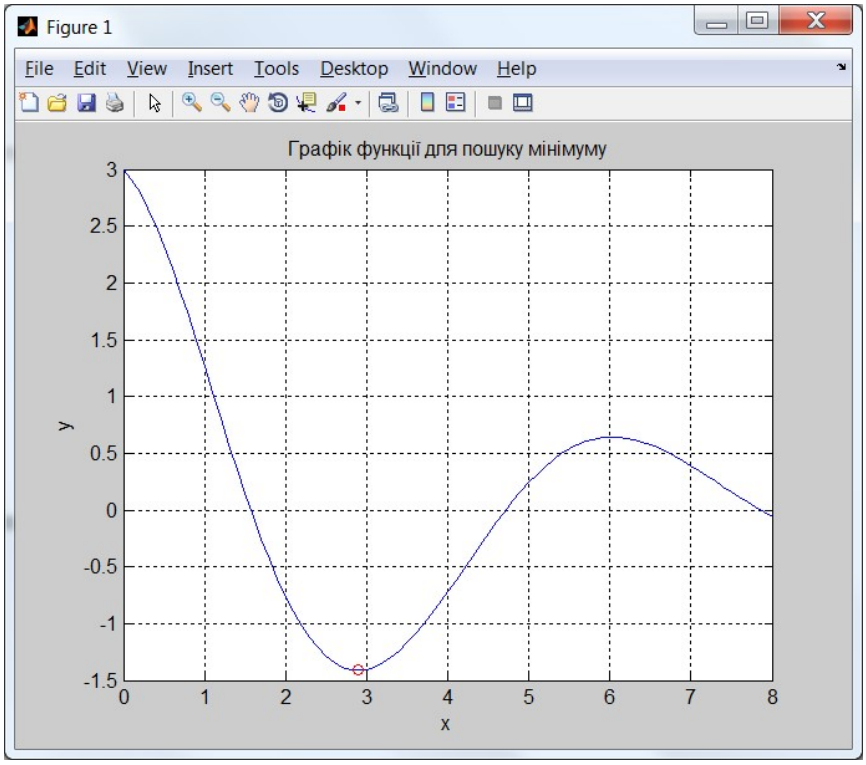

Рис. 5.1. Графік функції з точкою мінімуму

5.1.2 Мінімізація функцій кількох змінних

Дещо складніше завдання мінімізації функцій декількох змінних $f(x_1, x_2, ...)$. При цьому значення змінних представляються вектором *x*, причому початкові значення задаються вектором *x0*. Для мінімізації функцій ряду змінних MATLAB звичайно використовує різновиди симплекс-методу Нелдера-Міда. Цей метод є одним з кращих прямих методів мінімізації функцій ряду змінних, що не вимагає обчислення градієнта або похідних функції. Він зводиться до побудови симплекса в n-мірному просторі, заданого n+1 вершиною. В двовимірному просторі симплекс є трикутником, а в трьохмірному - пірамідою. На кожному кроці ітерацій вибирається нова точка рішення всередині або поблизу симплекса. Вона порівнюється з однією з вершин симплекса. Найближча до цієї точки вершина симплекса зазвичай замінюється цією точкою. Таким чином, симплекс перебудовується і зазвичай дозволяє знайти нове, більш точне положення точки рішення. Рішення повторюється, поки розміри симплекса по всім змінним не стануть менше заданої похибки рішення. Реалізує симплекс-методи Нелдера-Міда функція *fminsearch*, яка записується у вигляді:

- *fminsearch (fun, x0)* повертає вектор *x*, який є локальним мінімумом функції *fun(x)* поблизу *x0*. *x0* може бути скаляром, вектором (відрізком при мінімізації функції однієї змінної) або матрицею (для функції кількох змінних);

- *fminsearch (fun, x0, options)* аналогічна описаній вище функції, але використовує вектор параметрів *options* так само, як функція *fminbnd*;

- *fminsearch (fun, x0, options, P1, P2, ...)* схожа з описаною вище функцією, але передає в мінімізуючу функцію декількох змінних *fun(x, P1, P2, ...)* її додаткові аргументи *P1, P2*, ... Якщо потрібно використовувати параметри обчислень за замовчуванням, то замість *options* перед *P1, P2* необхідно ввести [];

- *[x, fval] = fminsearch (...)* додатково повертає значення цільової функції *fval* в точці мінімуму;

- *[x, fval, exitflag] = fminsearch (...)* додатково повертає параметр *exitflag*, що приймає додатнє значення, якщо процес ітерацій сходиться з використанням *options.tolX*, від'ємне — якщо ітераційний процес не сходиться до отриманого рішення *x*, та *0*, якщо перевищено максимальне число ітерацій *options.maxiter*;

- *[x, fval, exitflag, output] = fminsearch (...)* повертає структуру (запис) *output*;

- *output.algorithm* - використаний алгоритм;

- *output.funcCount* - число оцінок цільової функції;

- *output.iterations* - число проведених ітерацій.

Нехай потрібно знайти локальний мінімум функції

$$z(x,y) = \sin(3\pi x) \cdot \sin(3\pi y) \cdot e^{-x^2-y^2}$$

Для цього спочатку визначимо початкове наближення для пошуку, тобто отримаємо уявлення про поведінку функції. Побудуємо лінії рівня (рис…), що вкажуть на області пошуку локального мінімуму.

```
>> [X, Y]=meshgrid(0:0.02:1);
>> Z=sin(3*pi*X).*sin(3*pi*Y).*exp(-X.^2-Y.^2);
>> [CMatr,h]=contourf(X,Y,Z);
>> clabel(CMatr,h)
>> grid on
>> xlabel('x');
>> ylabel('y');
>> title('Лінії рівня');
```

З графіка видно, що локальні мінімуми знаходяться в околицях точок [0,5; 0,15], [0,2; 0,5] та ін.

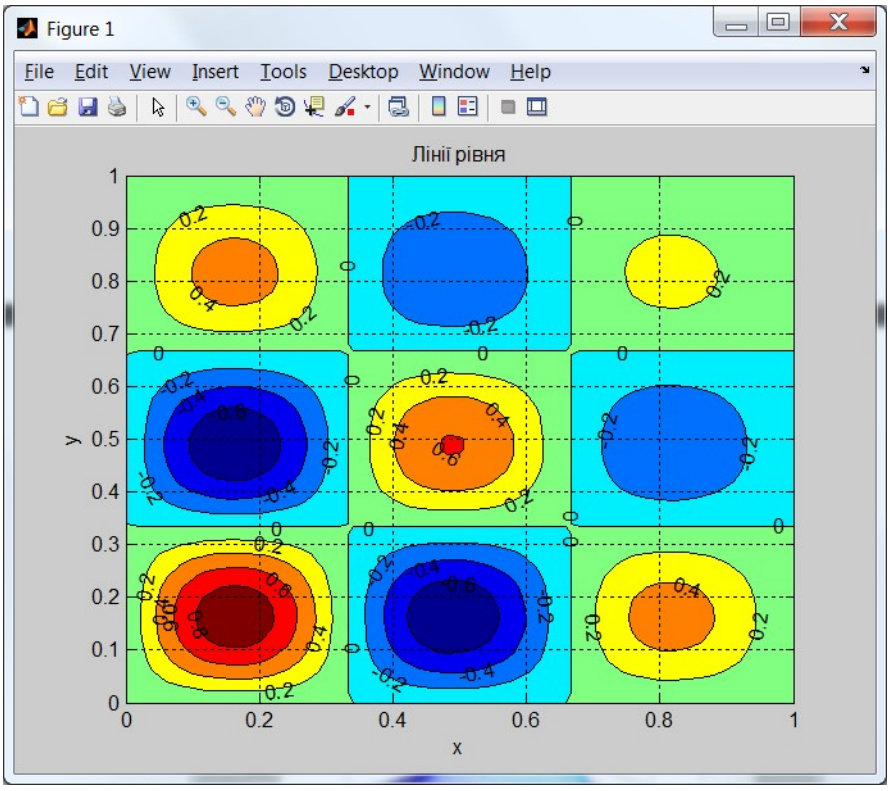

Рис. 5.2. Лінії рівня, орієнтовне розміщення локальних мінімумів та максимумів

Тепер сформуємо файл-функцію, яка розраховує значення шуканої функції. Аргументом файл-функції повинен бути вектор, перший елемент якого відповідає змінній *x*, а другий — змінній *y*.

```
function f=f2search(X)
f=sin(3*pi*X(1)).*sin(3*pi*X(2)).*exp(-X(1).^2-X(2).^2);
```

Замість файл-функції для розрахунків можна скористатися анонімною функцією:

```
>>f2search=@(x)sin(3*pi*x(1)).*sin(3*pi*x(2)).*exp(-x(1).^2-x(2).^2)
```

Тепер потрібно викликати функцію *fminsearch* з аргументами, що відповідають назві файл-функції та початковому наближенню. Результат роботи функції (вектор, компоненти якого є координатами локального мінімуму) присвоїмо змінній *res*.

```
>> res=fminsearch('f2search',[0.5, 0.15])
res =
    0.4891    0.1630
```

Для того, щоб отримати не тільки координати мінімуму, а й його значення, потрібно викликати функцію *fminsearch* з двома вихідними аргументами:

```
>> [res, Fmin]=fminsearch('f2search',[0.5, 0.15])
res =
    0.4891    0.1630
Fmin =
   -0.7621
```

Аналогічну процедуру слід виконати для інших локальних мінімумів.

5.2 Порядок виконання роботи

1.1. Для заданої функції на інтервалі *[a; b]* згідно варіанта *N* знайти всі локальні екстремуми. Побудувати графік функції, нанести всі отримані особливі точки на графік.

N	Функція	Інтервал
1	$f(x) = x^{0.5} + \sin(x) + 0.1x$	$x \in [0{,}1; 10{,}1]$
2	$f(x) = \sqrt{x} + \cos(x) + 0.2x$	$x \in [0{,}5; 8]$

N	Функція	Інтервал
3	$f(x)=\sin(x)+\cos(2x)+0.3x$	$x\in[-1{,}5;\ 9{,}1]$
4	$f(x)=\cos(x)+\sin(2x)+0.4x$	$x\in[-2{,}1;\ 7{,}5]$
5	$f(x)=\sin(x)+0.5\cos(2x)+0.5x$	$x\in[0{,}3;\ 8{,}7]$
6	$f(x)=\cos(x)+0.5\sin(2x)+0.6x$	$x\in[0{,}1;\ 10{,}1]$
7	$f(x)=e^{-0.3x}\sin(x)+0.7x$	$x\in[0{,}5;\ 8]$
8	$f(x)=e^{-0.3x}\cos(x)+0.8x$	$x\in[-1{,}5;\ 9{,}1]$
9	$f(x)=x^2\cos(2x)+0.9x$	$x\in[-2{,}1;\ 7{,}5]$
10	$f(x)=\sqrt{x}+\sin(3x)+x$	$x\in[0{,}3;\ 8{,}7]$
11	$f(x)=\sqrt{x}+\cos(1.8x)+1.1x$	$x\in[0{,}1;\ 10{,}1]$
12	$f(x)=x\cdot e^{-x}\sin(0.8x)+1.3x$	$x\in[0{,}5;\ 8]$
13	$f(x)=e^{-\sin(x)}+1.4x$	$x\in[-1{,}5;\ 9{,}1]$
14	$f(x)=e^{-\cos(x)}+1.5x$	$x\in[-2{,}1;\ 7{,}5]$
15	$f(x)=\ln(x)+1.1\sin(x)+1.6x$	$x\in[0{,}3;\ 8{,}7]$
16	$f(x)=\ln(x)+0.9\cos(x)+1.7x$	$x\in[0{,}1;\ 10{,}1]$
17	$f(x)=\ln(x)\sin(2x)+1.8x$	$x\in[0{,}5;\ 8]$
18	$f(x)=\ln(0.86x)\cos(0.75x)+1.9x$	$x\in[1{,}5;\ 6{,}1]$
19	$f(x)=1/x+\sin(0.56x)+2x$	$x\in[0{,}4;\ 5{,}5]$
20	$f(x)=1.23/x+\cos(1.44x)+2.1x$	$x\in[0{,}3;\ 4{,}7]$

1.2. Для заданої варіантом *N* функції на площині, обмеженій лініями x_1=-5, x_2=5, y_1=-5, y_2=5 знайти всі локальні екстремуми. Побудувати графік функції у вигляді ліній рівня та у тривимірному просторі, нанести всі отримані особливі точки на графік.

N	Функція
1	$z(x,y)=\sin(x)e^{-3y}$
2	$z(x,y)=\sin^2(x)\ln(y)$
3	$z(x,y)=\sin^2(x-2y)e^{-\lvert y\rvert}$
4	$z(x,y)=\dfrac{x^2y^2+2xy-3}{x^2+y^2+1}$
5	$z(x,y)=\dfrac{\sin(xy)}{x}$
6	$z(x,y)=(\sin(x^2)+\cos(y^2))^{xy}$
7	$z(x,y)=\cos^2(x)\ln(y)$
8	$z(x,y)=x^3\sin(xy)$

N	Функція
9	$z(x,y)=(x^2-2)\cos(y^2)$
10	$z(x,y)=(1+xy)(3-x)(4-y)$
11	$z(x,y)=e^{-\lvert x\rvert}(x^5+y^4)\sin(xy)$
12	$z(x,y)=(y^2-3)\sin\left(\frac{x}{\lvert y\rvert+1}\right)$
13	$z(x,y)=(x^3+y)\cos(e^y)$
14	$z(x,y)=\cos^2(x)\ln(y^2)$
15	$z(x,y)=\arctan(x+y)(\arccos(x)+\arcsin(y))$
16	$z(x,y)=\frac{\sin(xy)}{x}$
17	$z(x,y)=(1+xy)(3-x)(4-y)$
18	$z(x,y)=(y^2-3)\sin\left(\frac{x}{\lvert y\rvert+1}\right)$
19	$z(x,y)=\frac{x^2y^2+2xy-3}{x^2+y^2+1}$
20	$z(x,y)=x^3\sin(xy)$

5.3 Контрольні запитання

1. Задача пошуку екстремуму функції.
2. Пошук мінімуму функції однієї змінної.
3. Синтаксис та призначення аргументів функції fminbnd.
4. Вектор параметрів обчислень options.
5. Пошук екстремуму функції багатьох змінних.
6. Синтаксис та призначення аргументів функції fminsearch.
7. Визначення початкового наближення.

6 ЛАБОРАТОРНА РОБОТА №6
РОЗВ'ЯЗАННЯ ЗВИЧАЙНИХ ДИФЕРЕНЦІЙНИХ РІВНЯНЬ

Мета роботи: ознайомитись з вбудованими методами розв'язання диференційних рівнянь, отримати навички використання пакету «Symbolic Math».

6.1 Теоретичні відомості

6.1.1 Задача Коші

6.1.1.1 Функції вирішення диференційних рівнянь

Задача Коші для диференційного рівняння полягає в знаходженні функції яка задовольняє диференційному рівнянню довільного порядку:

$$y^{(n)} = f(t, y, y', ..., y^{(n-1)})$$

и початковим умовам при $t = t_0$

$$y(t_0) = u_0,\ y'(t_0) = u_1, \dots,\ y^{(n-1)}(t_0) = u_{n-1}$$

Для знаходження розв'язків звичайних диференційних рівнянь в Matlab є декілька спеціальних функцій, які в чисельних методах називаються *солверами*. Ці солвери реалізують різні чисельні методи.

У загальному вигляді виклик солвера виконується так:

[T, Y] = solver(odefun,interval, Y0,options)

де *solver* – відповідний солвер,

odefun – функція для обчислення вектор-функції правої частини системи рівнянь,

interval- масив з двох чисел, який задає відрізок для розв'язання рівняння,

Y0 – вектор початкових значень розшукуваної вектор-функції,

options – структура для керування параметрами і ходом обчислень.

Солвер повертає масив *T* з координатами вузлів сітки, в яких знайдено розв'язок, і матрицю *Y*, кожний стовпчик якої є значенням компоненти вектор-функції розв'язку у вузлах сітки.

Для розв'язування задачі Коші в Matlab є ряд солверів: *ode45,ode23, ode113, ode15s, ode23s, ode23t, ode23th*:

- *ode45* - однокрокові явні методи Рунге-Кутта 4-го і 5-го порядку. Це класичний метод, рекомендований для початкової проби рішення. У багатьох випадках він дає пристойні результати;

- *ode23* - однокрокові явні методи Рунге-Кутта 2-го і 4-го порядку. При помірній жорсткості системи ОДУ і низьких вимогах до точності цей метод може дати виграш в швидкості вирішення;
- *ode113* - багатокроковий метод Адамса-Башворта-Мултона змінного порядку. Це адаптивний метод, який може забезпечити високу точність рішення
- *ode23tb* - неявний метод Рунге-Кутта на початку розв'язання та метод, що використовує формули зворотного диференціювання 2-го порядку. Незважаючи на порівняно низьку точність, цей метод може виявитися більш ефективним, ніж *ode15s*;
- *ode15s* - багатокроковий метод змінного порядку (від 1 до 5, за замовчуванням 5), що використовує формули чисельного диференціювання. Це адаптивний метод, його варто застосовувати, якщо солвер ode45 не забезпечує вирішення;
- *ode23s* - однокроковий метод, що використовує модифіковану формулу Розенброка 2-го порядку. Може забезпечити високу швидкість обчислень при низькій точності рішення жорсткої системи диференціальних рівнянь;
- *ode23t* - метод трапецій з інтерполяцією. Цей метод дає добрі результати при вирішенні завдань, що описують коливальні системи з майже гармонійним вихідним сигналом.

Схема розв’язування в Matlab складається з таких етапів:

1. Приведення диференційного рівняння до системи диференційних рівнянь першого порядку (якщо зразу задано таку систему, то цей етап не потрібний).
2. Написання спеціальної функції (m-файлу) для системи рівнянь.
3. Виклик підходящого солвера.
4. Графічне відображення результатів.

6.1.1.2 М-функції

М-функції є М-файлами, які допускають наявність вхідних і вихідних аргументів. Вони працюють із змінними в межах власної робочої області, відмінної від робочої області системи MATLAB.

Загальний синтаксис m-функції

function [<перелік вихідних змінних>]=<ім’я-функції>(перелік вхідних змінних)

<тіло функції>

Важливо пам’ятати, що m-файл, що містить функцію, повинен називатися <ім’я-функції>.m .

Наприклад

```
function out1=func1(arg1, arg2)
out1=arg1+arg2;
```

Тут функція *func1* повертає суму (*out1*) аргументів *arg1* та *arg2*.

```
function [out1, out1]=func2(arg1, arg2)
out1=arg1+arg2;
out2=arg1-arg2;
```

Функція *func2* повертає суму (*out1*) аргументів *arg1* та *arg2* і, одночасно з цим, їх різницю (*out2*).

Використовувати створену m-функцію можна як і будь-яку стандартну.

```
>> [a,b]=func2(5,3)
a
=8
b
=2
```

Звичайно, що MATLAB повинен знати, де лежить створений файл з функцією, тому шлях до нього треба вказати в налаштуваннях програми (*File→Set Path…* в меню робочого середовища MATLAB).

Для прикладу знайдемо рішення рівняння

$y'(x) + 0.33y(x) = e^{-x}$, $y(0) = 0$ на інтервалі [0; 10].

Файл-функція для вирішення диференційного рівняння будується за певними правилами. Вона має два вхідних аргументи: змінну, по якій проводиться диференціювання, та вектор, розмір якого відповідає кількості невідомих функцій системи. Вихідним аргументом файл-функції є вектор правої частини системи.

Оскільки це рівняння першого порядку, немає потреби у приведенні до системи рівнянь.

Для рівняння $y'(x) + 0.33y(x) = e^{-x}$ файл-функція матиме вигляд

```
function F=rdif(x,y)
F=-0.33*y+2*exp(-x);
```

6.1.1.3 Виклик солвера

Тепер знайдемо рішення з заданими початковими умовами:

```
%вектор початкових умов
y0=0;
%виклик солвера
[T, Y]= ode45('rdif', [0 10], y0);
%побудова графіка знайденого рішення
plot(T,Y,'k-');
grid on
%оформлення графіка
xlabel('x');
ylabel('y');
```

```
title('Рішення рівняння {\ity} \prime+0.33 {\ity} = e^{-x}');
```

Графік знайденого рішення диференційного рівняння на заданому проміжку матиме вигляд, як на рис. 4.1. За допомогою формату синтаксису TeX можна оформити написи так, як це прийнято у математиці (так зроблено у функції *title*).

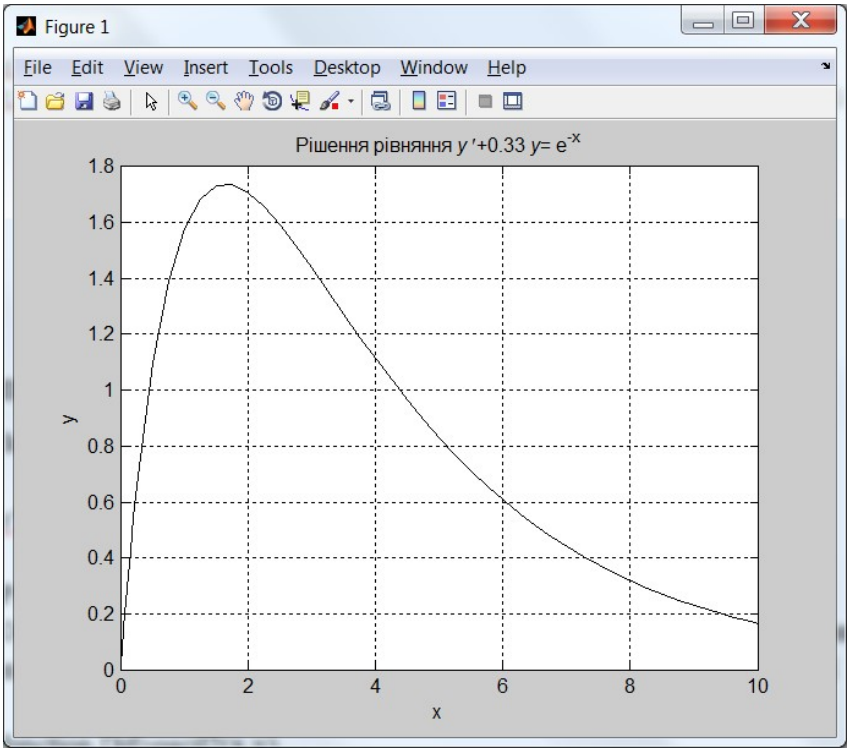

Рис. 6.1. Рішення диференційного рівняння

Дещо складніша процедура, у порівнянні з рішенням ЗДУ першого порядку, для вищих порядків.

Нехай потрібно знайти розв'язок диференційного рівняння другого порядку

$$y''(t)+2y'(t)+5y(t)=8$$

при початкових умовах $y(0)=1,\ y'(0)=0$ на інтервалі [0, 12].

1. Приведемо рівняння до системи рівнянь першого порядку. Для цього виконаємо заміну

$$\begin{cases} y_1 = y \\ y_2 = y' \end{cases}$$

Отримаємо систему рівнянь

$$\begin{cases} y_1' = y_2 \\ y_2' = -2y_2 - 5y_1 + 8 \end{cases}$$

$$\begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

2. Напишемо файл-функцію диференційного рівняння для цієї системи. Вона повинна мати два вхідних аргументи: змінну *t*, по якій виконується диференціювання (навіть якщо у явному вигляді ця змінна не використовується), і вектор, розмір якого дорівнює числу невідомих системи. Вихідним аргументом функції є вектор правої частини системи.

```
function F=rdif2(t,y)
%вектор компонентів системи
F=[y(2); -2*y(2)-5*y(1)+8];
```

1. Знайдемо рішення рівняння. Для цього використаємо солвер ode113. Його рішенням будуть два вихідні аргументи: вектор, що містить значення часу *t*, та матриця значень невідомих функцій у відповідні моменти часу. Так як зроблена заміна $y_1 = y, y_2 = y'$, то у першому стовпчику матриці знаходитиметься рішення рівняння y(t), а в другому — значення його похідної. Через те, що матриця результатів досить велика, відразу покажемо результат вирішення на графіках (рис. ***).

```
%вектор початкових умов
Y0=[1;0];
%виклик солвера від функції rdif2,
%початкового і кінцевого моменту часу і
%вектора початкових умов
[T,Y]=ode113(@rdif2,[0 12],Y0);
%побудова графіків
plot(T,Y(:,1),'k.-',T,Y(:,2),'b*-');
grid on
%оформлення графіків
xlabel('x');
ylabel('y, y \prime');
legend('Рішення', 'Швидкість');
title('Рішення рівняння {\ity} \prime\prime+2 {\ity}
\prime + 5 {\ity} = 8');
```

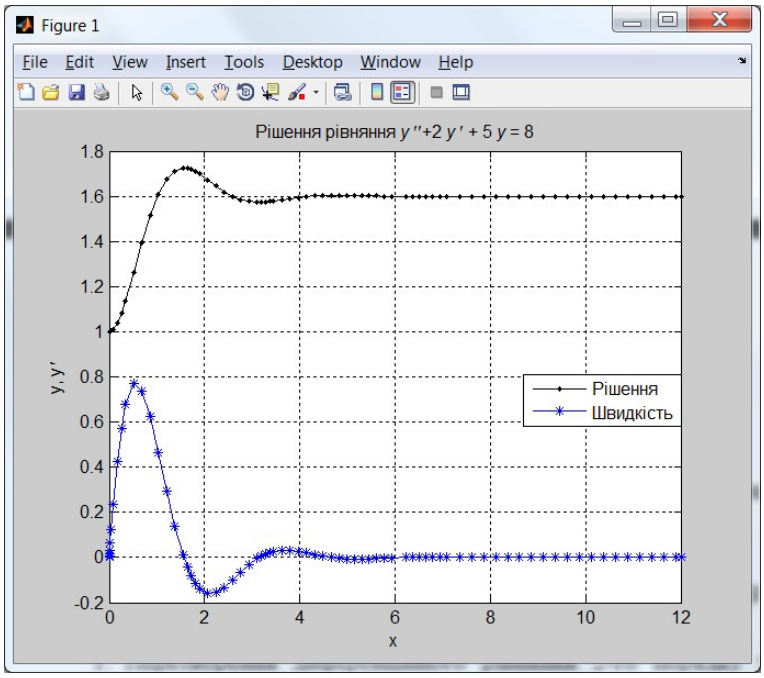

Рис. 6.2. Рішення ЗДУ другого порядку

6.1.1.4 Керування точністю пошуку рішення

Точність та похибка обчислень мають суттєвий вплив на якість отриманого наближеного рішення. Для керування роботою солверів, зокрема для встановлення точності, використовується додатковий параметр *options*, який потрібно сформувати функцією *odeset*.

options = odeset(... , вид контролю, значення, ...).

Повний список параметрів можна знайти в документації.

Наприклад, щоб зменшити величину відносної похибки (за замовчуванням 10^{-3}) до 10^{-4} потрібно сформувати структуру *options* у вигляді

```
options = odeset('RelTol', 1.0e-04)
```

а потім використати її як останній аргумент солвера.

6.1.1.5 Аналітичний вираз рішення ЗДУ

Для вирішення диференціальних рівнянь в символьному вигляді в MATLAB використовується функція *dsolve*, яка має наступні формати звернення:

1. *y = dsolve ('Dy(x)')*,

де Dy(x) — рівняння; y - рішення , що повертаються функцією *dsolve*.

2. *y = dsolve ('Dy (x)', 'ПУ')*,

де *Dy(x)* — рівняння; *ПУ* — початкові умови.

При використанні функції слід дотримуватись певних обмежень на форму запису рівняння (або системи рівнянь). Так, якщо невідома функція позначена символічною змінною *y*, то її похідні слід позначати як *D[n]y*, де в квадратних

дужках вказано порядок похідної. Таким чином, похідна повинна позначатися в символьному вираженні *Dy*, друга похідна - *D2y* і т.д. Функція *dsolve* може викликатися з різним набором параметрів,в залежності від типу розв'язуваної задачі і порядку системи рівнянь. Деталі можна знайти в документації.

Загальне рішення рівняння

$y'(x) + 0.33y(x) = e^{-x}$

матиме вигляд

```
>> syms y;
>> S=dsolve('Dy=-0.33*y+exp(-x)','x')
S =
C4/exp((33*x)/100)-100/(67*exp((33*x)/100)*exp((67*x)/100))
```

або, використавши функцію pretty для представлення символьних виразів у легшоми для сприйняття виді

```
>> pretty(S)
      C4                    100
  ----------- - -------------------------
     / 33 x \          / 33 x \    / 67 x \
  exp| ---- |    67 exp| ---- | exp| ---- |
     \ 100  /          \ 100  /    \ 100  /
```

Останнім аргументом функції *dsolve* вказаний *x* як змінна для диференціювання (за замовчуванням використовується змінна *t*).

З початковими умовами $y(0) = 0$ рішення буде знайдено як:

```
>> syms y;
>> S=dsolve('Dy=-0.33*y+exp(-x)','y(0)=0','x')
S =
100/(67*exp((33*x)/100)) - 100/(67*exp((33*x)/100)*exp((67*x)/100))
>> pretty(S)
       100                    100
  -------------- - -------------------------
       / 33 x \          / 33 x \    / 67 x \
  67 exp| ---- |    67 exp| ---- | exp| ---- |
       \ 100  /          \ 100  /    \ 100  /
```

6.1.2 Краєві задачі для звичайних диференційних рівнянь

6.1.2.1 Постановка задачі

Необхідно знайти функцію *y(x),* яка задовольняє на відрізку *[a,b]* диференційному рівнянню

$$y^{(n)} = f(x, y, y', ..., y^{(n-1)})$$

і граничним умовам (у даному випадку лінійним)

$\alpha_0 y(a) + \alpha_1 y'(a) = A, \quad \beta_0 y(b) + \beta_1 y'(b) = B.$

Розв'язання задачі складається з наступних етапів.

1. Перетворення диференційного рівняння у систему рівнянь 1-го порядку.
2. Написання функції для обчислення правої частини системи.
3. Написання функції, яка визначає граничні умови.
4. Формування початкового наближення за допомогою функції *bvpinit*.
5. Виклик солвера *bvp4c* для вирішення граничної задачі.
6. Графічна інтерпретація результатів.

6.1.2.2 Рішення задачі

Розглянемо процедуру рішення на прикладі рівняння

$y'' - y'(4 + 2y/x) = x/2$ на відрізку $[1{,}3;\ 1{,}6]$ при граничних умовах

$1{,}5y(1{,}3) - y'(1{,}3) = 0{,}6, \quad 2y(1{,}6) = 0{,}3$

2. Зведемо рівняння до системи виду:

Використавши заміну (як і в задачі Коші)

$$\begin{cases} y_1 = y \\ y_2 = y_1' \end{cases},$$

отримаємо систему рівнянь

$$\begin{cases} y_1' = y_2 \\ y_2' = y_2(4 + 2y_1/x) + x/2 \end{cases},$$

а граничні умови для допоміжних функцій перетворимо так, щоб в правій частині були 0:

$$\begin{cases} 1{,}5y(1{,}3) - y'(1{,}3) - 0{,}6 = 0 \\ 2y(1{,}6) - 0{,}3 = 0 \end{cases}$$

3. Створимо функцію для обчислення правих частин системи рівнянь rdifbound

```
%функція rside для правих частин системи рівнянь
function F=rdifbound(x,y)
F=[y(2); y(2)*(4+2*y(1)/x)+x/2];
```

Збережемо функцію в m-файлі з тим самим іменем.

Ця функція має два вхідні аргументи *x, y*.

Перший — значення *x*, другий – вектор *y*, що складається з двох елементів: *y(1)* відповідає y_1 , а *y(2)* змінній y_2.

4. Напишемо функцію для завдання граничних умов, *bound:*

```
%функція bound граничних умов
function F=bound(ya,yb)
F=[1.5*ya(1)-ya(2)-0.6; 2*yb(1)-0.3];
```

Вона також має два вхідні аргументи, *ya, yb* для першої і другої умови.

5. Формування початкового наближення.

Солвер *bvp4c* базується на методі скінчених різниць, тобто, отриманий розв'язок є вектори значень невідомих функцій в точках відрізка (у вузлах рівномірної сітки). Вибір початкової сітки і наближення може впливати на якість розв'язку. Для формування початкової сітки і наближення призначена функція *bvpinit*.

Ця функція має такий загальний вигляд:

initsol = bvpinit (початкова сітка, початкове наближення до рішення)

Початкова сітка визначається вектором координат вузлів, впорядкованих за зростанням і розташованих на відрізку *[a,b]*. Формування рівномірної сітки виконується за допомогою функції *linspace*, яка має такий загальний вигляд:

meshinit =linspace(a,b,n),
де *a,b* – граничні точки відрізку, *n* – кількість точок.

Для нашого прикладу ця функція має вигляд:

```
>> meshinit = linspace(1.3, 1.6, 20);
```

Початкове наближення до рішення задається вектором з двох елементів для функцій y_1 і y_2.

```
>> yinit = [1 0];
```

Тепер можна сформувати структуру *initsol* з інформацією про початкове наближення функцією *bvpinit*:

```
>> initsol = bvpinit(meshinit, yinit);
```

6. Виклик солвера *bvp4c* для вирішення граничної задачі.

Вхідними аргументами солвера є вказівники на функції правої частини системи і граничних умов. При необхідності, можна задати параметри управління обчисленнями.

В результаті роботи *bvp4c* формується структура *sol* з інформацією про розрахункову сітку, значення розшукуваної функції і її похідної.

```
>> sol = bvp4c(@rside, @bound, initsol)
```

Результатом виконання функції *bvp4c* є структура

```
sol =

    solver: 'bvp4c'
         x: [1x20 double]
         y: [2x20 double]
        yp: [2x20 double]
     stats: [1x1 struct]
```

де в *sol.x* знаходяться координати сітки; в *sol.y* — матриця, в якій записуються значення функції *y(x)* і її похідної вигляду:

перший рядок *sol.y(1, :)* відповідає значенням функції y_1 в точках сітки *sol.x(:)*

другий рядок *sol.y(2, :)* відповідає значенням функції y_2 в *sol.x(:)*

7. Графічна інтерпретація результатів

Виведемо графік результатів обчислень.

```
>> plot(sol.x, sol.y(1, :), 'r.-')
>> hold on
>> plot(sol.x, sol.y(2, :), 'k.')
>> title('Розв''язок граничної задачі солвером bvp4c')
>>  legend('Значення функції y1 в сітці sol.x','Значення функції y2 в сітці sol.x',4)
>> xlabel('x')
>> ylabel('y')
>> grid on
>> hold off
```

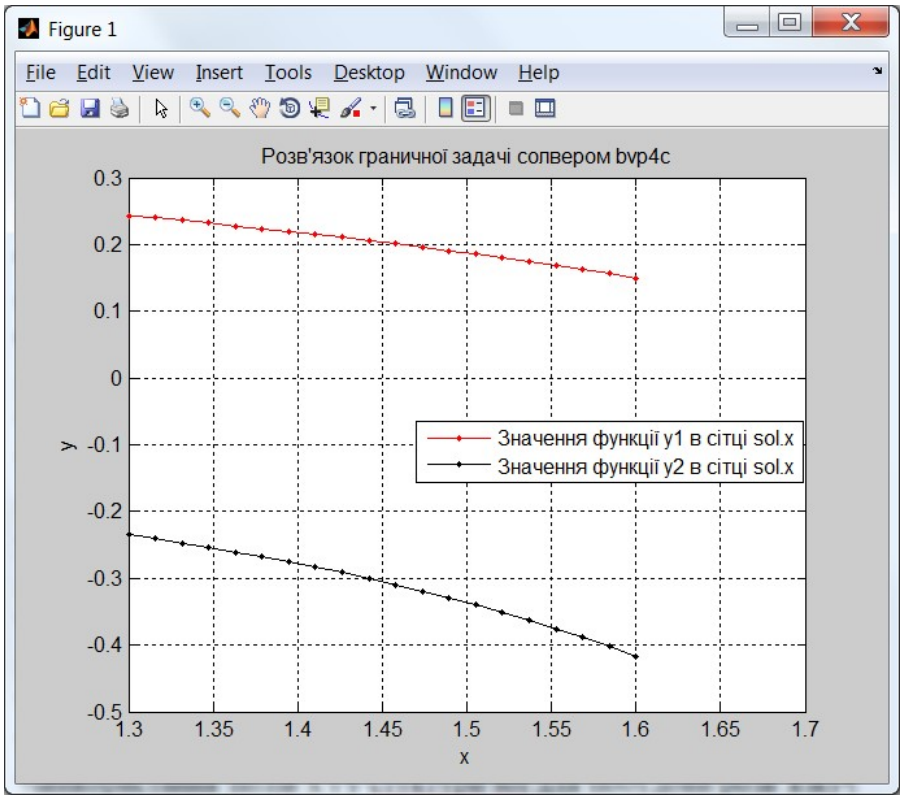

Рис. 6.3. Рішення граничної задачі

6.2 Порядок виконання роботи

4.1. Знайти розв’язок звичайного диференційного рівняння першого порядку, зобразити графік результатів (задача Коші).

№	Диференційне рівняння
1	$2xy'(x+1)-1=0,\quad y(1)=-1$
2	$y'+2xy=e^{-x^2},\quad y(0)=-1$
3	$xy'=y+\sqrt{xy},\quad y(e)=e$
4	$y'+x^{-2}y=e^{1/x},\quad y(-1)=0$
5	$xy'=y-xe^{1/x},\quad y(1)=0$
6	$xy'-2y=x^3\cos x,\quad y(\pi/6)=0$
7	$2yy'(1+e^x)=e^{2x},\quad y(0)=1$
8	$x(y'-y)=(x+1)e^x,\quad y(1)=e$
9	$2y'x^2=(x+y)^2,\quad y(1)=1$
10	$(y'x-y)\cos^2 x=x^2,\quad y(\pi/4)=0$
11	$2xyy'=y^2-x^2,\quad y(1)=1$
12	$x^2y'=y^2e^{-1/y},\quad y(-1)=0.5$

№	Диференційне рівняння
13	$(y'-y)(x^2+1)=e^x, \quad y(1)=0$
14	$x^2y'=2xy+y^2+xy+x^2, \quad y(1)=0$
15	$2xyy'=x^2+3y^2, \quad y(1)=1$
16	$xy'-y=2x(x+1)/(x+2), \quad y(-1)=0$
17	$y'=(2xy+y^2-x^2)/(2x^2), \quad y(-1)=0$
18	$2y'x^2=(x+y)^2, \quad y(1)=1.5$
19	$y'+x^{-2}y=e^{1/x}, \quad y(-1)=0.5$
20	$2xyy'=y^2-x^2, \quad y(1)=2.1$

4.2. Знайти розв'язок диференційного рівняння на інтервалі *[0; 1]*, зобразити графік результатів (задача Коші). Початкові умови – нульові.

№	Диференційне рівняння
1	$y'''x-y''=x^2e^x$
2	$2xy'y''=x^2+3(y')^2$
3	$x^2y'''+1=0$
4	$xy'''+y''=9x^2$
5	$2x^2y''=2xy'+(y')^2-x^2$
6	$y'''(1+x^2)^2+2x=0$
7	$y''x+y'=x^2\ln x$
8	$x^2y''=2xy'+(y')^2$
9	$2y''\sqrt{x}=\ln x+2$
10	$xy'''-2y''=4x(x^2-1)$
11	$2(y')^2=y''(y-1)$
12	$xy'''-y''=x\cos x$
13	$y''-2(y')^2ctgy=0$
14	$yy''=(y')^2+2y'$
15	$y'''(x+1)=x+2$
16	$2y(y')^2=y''(y^2-1)$
17	$y''(e^x+1)+(y')^2=0$
18	$y'''(x^2+1)+2xy''=0$
19	$y''+(y')^2=y'$

№	Диференційне рівняння
20	$y''(y^2+1)=2y(y')^2$

4.3. Знайти розв'язок граничної задачі для диференційного рівняння та зобразити графік результатів. N — номер варіанта.

$$y''=-0.3N\sin(x+N/10),\quad y(0)=0,\ y'((N+5)\pi/2)+0.2Ny((N+5)\pi/2)=-1$$

6.3 Контрольні запитання

1. Схема рішення задачі Коші для диференційного рівняння першого порядку.
2. Схема рішення задачі Коші для диференційного рівняння вищих порядків.
3. Схема рішення задачі Коші для системи диференційних рівнянь.
4. Схема рішення граничної задачі.
5. Солвери MATLAB.
6. Параметри солверів, аргументи, які повертають солвери.
7. Функції рішення граничної задачі.
8. Рішення диференційного рівняння у символьному вигляді.

7 ЛАБОРАТОРНА РОБОТА №7
ОСНОВИ ПРОГРАМУВАННЯ В MATLAB

Мета роботи: ознайомитись з механізмом m-функцій та його застосуванням для вирішення практичних задач, отримати навички створення m-функцій та роботи з ними.

7.1 Теоретичні відомості

7.1.1 Загальні відомості

7.1.2 Оператор if

Для того, щоб мати можливість реалізувати логіку, в програмі використовуються логічні оператори. Дані оператори можна представити у вигляді вузлових пунктів, досягаючи яких програма робить вибір за яким з можливих напрямків рухатися далі. Наприклад, потрібно визначити, чи містить деяка змінна *arg* додатне або від'ємне значення і вивести відповідне повідомлення на екран. Для цього можна скористатися оператором *if* (якщо), який і виконає потрібні перевірки. У самому простому випадку синтаксис даного оператора if має вигляд:

```
if <вираз >
<оператори >
end
```

Якщо значення параметра «вираз» відповідає значенню «Істина», то виконуються «оператори», інакше вони пропускається програмою. Слід зазначити, що «вираз» є умовним виразом, в якому виконується перевірка деякої умови. Результат виразу повинен давати відповідь «Істина» (true) чи «Хибність» (false). Для реалізації складених умов в MatLab використовуються логічні оператори: & - логічне «І», | - логічне «АБО», ~ - логічне «НЕ». Для зміни порядку виконання операцій (порядку обчислення виразу) використовуються круглі дужки.

Нижче наведений приклад реалізації функції *sign()*, яка повертає +1, якщо число більше нуля, -1 - якщо число менше нуля і 0, якщо число дорівнює нулю:

```
function my_signx = 5;
if x> 0
disp (1);
end
```

```
if x <0
disp (-1);
end
if  x == 0
disp (0);
end
```

Аналіз наведеного прикладу показує, що всі ці три умови є взаємовиключними, тобто при спрацьовуванні одного з них немає необхідності перевіряти інші. Реалізація саме такої логіки дозволить збільшити швидкість виконання програми. Цього можна домогтися шляхом використання конструкції:

```
if <вираз>
 <Оператори1>% виконуються, якщо істинна умова
else
<Оператори2>% виконуються, якщо умова хибна
End
```

Тоді наведений вище приклад можна записати наступним чином:

```
if x>0
    disp (1);
else
    if x <0
        disp (-1);
    else
        disp (0);
    end
end
```

У цій програмі спочатку виконується перевірка на позитивність змінної *x*, і якщо це так, то на екран виводиться значення 1, а всі інші умови ігноруються. Якщо ж перша умова виявилася хибною, то виконання програми переходить по *else* (інакше) на другу умову, де виконується перевірка змінної *x* на заперечність, і в разі істинності умови, на екран виводиться значення -1. Якщо обидві умови виявилися помилковими, то виводиться значення 0.

Наведений вище приклад можна записати в більш простій формі, використовуючи ще одну конструкцію оператора *if* мови MATLAB:

```
if <вираз1>
<Оператори1>% виконуються, якщо вираз істинний
```

```
    elseif <вираз2>
     <Оператори2>% виконуються, якщо істинний вираз 2
    ...
    else
     <ОператориN>% виконуються, якщо жоден з виразів не
істинний
    end
```

і записується таким чином:

```
if x> 0
    disp (1);% виконується, якщо x> 0
elseif x <0
    disp (-1);% виконується, якщо x <0
else
    disp (0);% виконується, якщо x = 0
end
```

За допомогою умовного оператора *if* можна виконувати перевірку більш складних (складених) умов. Наприклад, необхідно визначити: чи потрапляє змінна *x* у діапазон значень від 0 до 2? Це можна реалізувати одночасною перевіркою відразу двох умов: *x> = 0* і *x <= 2*. Якщо ці обидві умови істинні, то x потрапляє в діапазон від 0 до 2.

7.1.3 Оператор множинного вибору switch

У деяких завданнях програмування потрібно виконувати перевірку на рівність деякої змінної константним значенням. Наприклад, потрібно перетворити малі літери у великі. У цьому випадку необхідно зробити перевірку поточного символу з усіма можливими літерами алфавіту і при рівності з однією з них, замінити її на велику. Для вирішення таких завдань зручно користуватися оператором *switch*, який має наступний синтаксис:

```
switch <вираз>
case <значення виразу 1>,
    <Оператори1>
case{<значення виразу 2>,<значення виразу 2>,<значення
виразу 3>,...}
    <Оператори2>
    ...
otherwise,
    <Оператори інкаше>
end
```

Тут <*вираз*> — змінна, значення якої перевіряється на рівність тим чи іншим константам; <*значення виразу N*> — константи, з яким порівнюється значення змінної; *otherwise* — ключове слово для виконання операторів (<*Оператори інакше*>) при хибності всіх умов. Наведемо приклад роботи даного оператора для перетворення малих букв латинського алфавіту в заголовні.

```
function upper_symbol
    ch = 'c';
    switch ch
        case 'a', ch = 'A';
        case 'b', ch = 'B';
        case 'c', ch = 'C';
        case 'd', ch = 'D';
        case 'e', ch = 'E';
            ...
        case 'z', ch = 'Z';
    end
    disp (ch);
```

Тут задається символьна змінна *ch* зі значенням *c*. Потім, за допомогою оператора *switch*, перевіряється її значення з усіма можливими малими літерами латинського алфавіту від *a* до *z*. Як тільки одна з умов спрацювала, оператор *switch* завершує свою роботу і виконання програми переходить на функцію *disp()*, яка відображає значення змінної *ch* на екран.

7.1.4 Оператор циклу for

Часто при організації циклу потрібно перебирати значення лічильника в заданому діапазоні значень і з заданим кроком зміни. Наприклад, щоб перебрати елементи вектора (масиву), потрібно організувати лічильник від 1 до *N* з кроком 1, де *N* - число елементів вектора. Щоб обчислити суму ряду, також задається лічильник від *a* до *b* з необхідним кроком зміни *step*. Для простої реалізації таких операцій використовується оператор циклу *for*, який дозволяє простіше і наочніше реалізовувати цикл з лічильником.

Синтаксис оператора циклу *for* має наступний вигляд:

for <лічильник> = <початкове значення>: <крок>: <кінцеве значення>
<Оператори циклу>
end

Розглянемо роботу даного циклу на прикладі реалізації алгоритму пошуку максимального значення елемента у векторі:

```
function search_max
```

```
    a = [3 6 5 3 6 9 5 3 1 0];
    m = a (1);% поточне максимальне значення
    for i = 1: length (a)% цикл від 1 до кінця вектора з
кроком 1(за замовчуванням)
    if m <a (i)  % якщо a (i)> m, то m = a (i)
    m = a (i);
    end
    end % кінець циклу for
    disp (m);
```

У даному прикладі цикл *for* задає лічильник *i* ,що змінює своє значення від 1 до 10 з кроком 1. Зверніть увагу, що якщо величина кроку не вказується явно, то він береться за замовчуванням рівним 1. У наступному прикладі розглянемо реалізацію алгоритму зміщення елементів вектора вправо, тобто передостанній елемент ставиться на місце останнього, наступний - на місце передостаннього, і т.д. до першого елемента:

```
    function queue
    a = [3 6 5 3 6 9 5 3 1 0];
    disp (a);
    for i = length (a): -1:2% цикл від 10 до 2 з кроком -
1
         a (i) = a (i-1); % зміщуємо елементи вектора a
    end % кінець циклу for
    disp (a);
```

В результаті роботи даного циклу буде виведено:

```
3 6 5 3 6 9 5 3 1 0
3 3 6 5 3 6 9 5 3 1
```

Наведений приклад показує, що для реалізації циклу з лічильником від більшого значення до меншого, потрібно явно вказувати крок, в даному випадку, -1. Якщо цього не зробити, то цикл відразу завершить свою роботу і програма буде працювати некоректно.

7.1.5 Оператор циклу while

У найпростішому випадку цикл в програмі організовується за допомогою оператора *while*, який має наступний синтаксис:

```
while <умова>
    <оператори>
end
```

Тут <*умова*> означає умовний вираз подібне тому, яке застосовується в операторі *if*, цикл *while* працює до тих пір, поки ця умова вірна. Слід звернути увагу на те, що якщо умова буде помилковою до початку виконання циклу, то оператори, що входять в цикл, не будуть виконані жодного разу. Наведемо приклад роботи циклу *while* для підрахунку суми ряду $S = \sum_{i=1}^{20} i$:

```
function sum_i
S = 0;% початкове значення суми
i = 1;% лічильник суми
while i <= 20% цикл (працює поки i <= 20)
S = S + i;% підраховується сума
i = i +1;% збільшується лічильник на 1
end% кінець циклу
disp (S);% відображення суми 210 на екрані
```

7.2 Порядок виконання роботи

У відповідності до варіанта написати програму, що виконує поставлене завдання. Вихідні дані для роботи програми повинні задаватися користувачем. Результати роботи програми у зрозумілій формі повинні виводитися на екран. Текст програми повинен мати необхідні пояснення.

1. Записати кожен другий елемент цілочисельного масиву $X = (X_1, X_2, ..., X_n)$ поспіль в масив $Y = (Y_1, Y_2, ..., Y_k)$. Визначити кількість простих чисел в кожному масиві. Обчислити середнє арифметичне всіх елементів масивів X і Y.
2. Визначити максимальний елемент серед додатних непарних елементів і мінімальний серед додатних парних елементів цілочисельного масиву $X = (X_1, X_2, ..., X_n)$. Видалити з масиву всі прості числа, вивести повідомлення, скільки елементів було видалено.
3. Визначити, чи містить заданий масив групи елементів, розташовані в порядку зростання їх значень. Якщо так, то визначити кількість таких груп. Видалити з масиву першу таку групу.
4. У масиві цілих чисел *X(k)* поміняти місцями перший і мінімальний елементи. Видалити всі прості елементи, що стоять після максимального елемента. Знайти середнє арифметичне елементів масиву до і після видалення.

5. Обчислити середнє арифметичне елементів масиву $X = (X_1, X_2, ..., X_n)$, розташованих між його мінімальним і максимальним значеннями. Якщо мінімальний елемент розміщується в масиві раніше максимального, то упорядкувати масив на даному проміжку по зростанню його елементів, і навпаки, якщо мінімальний елемент розміщується після максимального, то впорядкувати за спаданням.
6. Визначити порядкові номери і значення першого додатного і останнього від'ємного елементів цілочисельного масиву *X(n)*. Визначити середнє арифметичне елементів масиву, позиційно розташованих між знайденими елементами. Передбачити випадок, що масив може не містити позитивних чи негативних елементів.
7. Задано масив *Z(n)* цілих чисел. Видалити з масиву найбільший і найменший елементи. У перетвореному масиві знайти середнє арифметичне семи найбільших елементів.
8. Задано масив *Y(k)* цілих чисел. Якщо він впорядкований, залишити його без зміни. Якщо масив не впорядкований, то вставити після кожного другого елементу мінімальне непросте число в масиві. Передбачити випадок, що масив складається тільки з простих чисел.
9. Заданий масив *Y(k)* цілих чисел. Визначити в масиві кількість простих двозначних чисел. Якщо таких чисел більше двох, видалити їх з масиву. Перевірити, чи змінився максимальний елемент масиву.
10. Дано масив дійсних чисел $X = (X_1, X_2, ..., X_n)$. Записати елементи заданого масиву X в масив Y наступним чином: в початковій частині розташувати додатні елементи в порядку зростання, потім у порядку убування від'ємні елементи, нульові елементи не записувати. Оцінити, як при цьому змінилися позиції максимального і мінімального елементів масиву.
11. У масиві цілих чисел *X(k)* поміняти місцями перший і мінімальний елементи. Видалити всі прості елементи, що стоять після максимального елемента. Знайти середнє арифметичне елементів масиву до і після видалення.
12. Задана матриця *A(n,n)*. Дзеркально відобразити її відносно головної діагоналі. У перетвореній матриці знайти рядки, елементи якої утворюють зростаючу послідовність.
13. Задана матриця *B(n,m)*. Визначити кількість стовпців, впорядкованих за зростанням. Кожен другий стовпець впорядкувати за зростанням і знайти, на скільки збільшилася кількість таких впорядкованих стовпців.

14. У матриці *X(n,n)* поміняти місцями елементи на головній та побічній діагоналях. Перевірити, чи змінилося положення максимального елемента в кожному рядку.
15. Задана матриця *A(n,n)*. Дзеркально відобразити її відносно побічної діагоналі. У перетвореній матриці знайти стовпчики, елементи яких утворюють послідовність, яка спадає. Підрахувати їх кількість та вивести їх на екран.
16. Визначити номери рядка і стовпця максимального елемента прямокутної матриці *A(n,m)*. Поміняти місцями перший і максимальний елементи матриці. Підрахувати кількість нульових елементів матриці і надрукувати їх індекси.
17. Знайти суму елементів квадратної матриці *X(n,n)*, що знаходяться по периметру цієї матриці і суму елементів на її діагоналях. Якщо суми рівні, то кожен від'ємний елемент матриці замінити модулем цього ж елемента.
18. Задана матриця *A(n,m)*, в кожному стовпці якої мінімальний елемент необхідно замінити сумою додатних елементів цього ж стовпця.
19. Задана матриця *A(n,n)*. Визначити максимальний елемент серед елементів матриці, розташованих вище головної діагоналі, і мінімальний елемент серед тих, що знаходяться нижче головної діагоналі. Якщо ці елементи рівні, знайти кількість таких чисел в матриці.
20. У матриці *D(n,m)* знайти і вивести номери стовпців, впорядкованих за спаданням. У кожному стовпці знайти кількість і суму позитивних елементів.

7.3 Контрольні запитання

1. Структура програми в MATLAB.
2. Умовний оператор if.
3. Оператор множинного вибору switch.
4. Оператор циклу for.
5. Оператор циклу while.

НАВЧАЛЬНО-МЕТОДИЧНІ МАТЕРІАЛИ

1. Ануфриев И. Е. Самоучитель MATLAB 5.3/6.x / И.Е. Ануфриев. – СПб.: БХВ-Петербург, 2004 – 736 с., ил.
2. Дьяконов В.П. MATLAB 6.5 SP1/7.0. Simulink 5/6. Основы применения. / В.П. Дьяконов — СПб.: СОЛОН-Пресс, 2005 – 576 с.: ил.;
3. Ануфриев И. Е. MATLAB 7 / И.Е. Ануфриев, А.Б. Смирнов, Е.Н. Смирнова – СПб.: БХВ-Петербург, 2005 – 1104 с., ил.
4. Лазарев Ю. Моделирование процесов и систем в MATLAB. Учебный курс / Ю. Лазарев – СПб. : Питер, Киев: BHV, 2005. – 512 с.: ил.
5. http://matlab.exponenta.ru/matlab/default.php

ДОДАТОК А. ПРАВИЛА ВИКОНАННЯ РОБОТИ ТА ОФОРМЛЕННЯ ПРОТОКОЛІВ

Робота виконується студентом самостійно, з використанням необхідної довідникової літератури та програмного забезпечення.

Результати виконання роботи студент оформлює у вигляді протоколу, що містить наступні розділи:

1. титульний аркуш;
2. зміст протоколу;
3. завдання до роботи;
4. короткі теоретичні відомості;
5. хід виконання роботи, що містить необхідні для розуміння та аналізу дані, включно з додатковими математичними залежностями, програмним кодом, блок-схемами, ілюстраціями тощо;
6. висновки.

Виконана робота захищається у вигляді співбесіди, під час якої студент демонструє елементи роботи, використання отриманих знань та навичок у відповідях на питання та виконанні практичних завдань.